PROCESS

PROCESS

- Windows proses leri iki parçadan oluşmaktadır.
  - İşletim sisteminin prosesi kontrol edip yönetmesini sağlayan çekirdek nesnesi,
  - Çalıştırılacak olan bütün kodu ve veriyi içerisine alacak bir adres alanı

Çekirdek Nesneleri(Kernel Objects)

- İşletim sisteminin temel araçlarını kullanması için oluşturduğu yapıdır.
- Örneğin Prosesler, iplikler (threads),I/O,mutex vb.
- Çekirdek nesneleri sadece çekirdek tarafından kullanılabilir. Uygulamalar çekirdek nesnelerine direkt olarak ulaşmaları imkansızdır.
- Kullanıcılar işletim sisteminin sağladığı API fonksiyonlarını kullanarak çekirdek nesneleri ile iletişime geçebilirler
- Bütün çekirdek nesneleri CreateXXX şeklinde başlayan fonksiyonlar ile oluşturulurlar.
- Bu fonksiyonlar dönüş değeri olarak oluşturulan çekirdek nesnesinin handle (kimlik) değerini döndürür.

Çekirdek Nesnesi Kullanım Sayısı

- Çekirdek nesnelerini kullanan proses sayısı işletim sistemi tarafından tutulur.
- Nesneyi kullanan bir proses kalmadığında işletim sistemi tarafından nesnenin kullandığı hafıza alanı temizlenir.
- Kullanım sayısı bütün çekirdek nesnelerinde bulunur.

Güvenlik

- Çekirdek nesneleri security descriptor adı verilen yapılar sayesinde koruma altına alınabilirler.
- Bu yapı çekirdek nesnesini kimin yarattığını ,kimin kullanabileceğini , kimin okuyabileceği ve kimin kullanamayacağını belirtir.
- Çekirdek nesnelerini oluşturan fonksiyonların hepsinde bir parametre ortaktır.
- Bu parametre aşağıdaki gibi bir yapının adresini oluşturur.

```
typedef struct _SECURITY_ATTRIBUTES {
   DWORD nLength;
   LPVOID lpSecurityDescriptor;
   BOOL bInheritHandle;
} SECURITY_ATTRIBUTES;
```

Kalıtım

- Çekirdek nesneleri prosesler arasında paylaştırılabilir.
- Tek bir şartla, prosesler arasında ebeveyn çocuk ilişkisi olmalıdır.
- Çekirdek nesneleri oluşturulurken güvenlik parametresin de nesnenin kalıtım yolu ile aktarılıp aktarılmayacağı belirtilir.

```
SECURITY_ATTRIBUTES sa;
sa.nLength = sizeof(sa);
sa.lpSecurityDescriptor = NULL;
sa.bInheritHandle = TRUE;   // Make the returned handle inheritable.

HANDLE hMutex = CreateMutex(&sa, FALSE, NULL);
```

- Eğer programcı isterse çekirdek nesnelerinin kalıtım özelliğini değiştirebilir.

```
BOOL SetHandleInformation(
   HANDLE hObject,
   DWORD dwMask,
   DWORD dwFlags);
```

```
SetHandleInformation(hobj, HANDLE_FLAG_INHERIT, HANDLE_FLAG_INHERIT);
```

Çekirdek nesnelerinin sonlandırılması

- Çekirdek nesneleri aşağıdaki fonksiyon kullanılarak sonlandırılabilir.

```
BOOL CloseHandle(HANDLE hobj);
```

- Çekirdek nesneleri proseslere değil çekirdeğe aittir.
- Bir prosesin sonlanması kullandığı bütün çekirdek nesnelerinin de kaldırılacağı anlamına gelmez.
- Nesneyi kullanan başka bir proses olabilir.
- Eğer çekirdek nesnesini kullanan proses sayısı sıfır olursa çekirdek bu nesneyi sistemden kaldırır.

Proses Çekirdek Nesne Tablosu

- Bir proses oluşturulduğunda sistem ona bir adet handle(kimlik) tablosu oluşturur.
- Bu tablo sadece çekirdek nesneleri içindir. Kullanıcı ara yüzleri ve GDI nesneleri burada yer almaz.
- Bu tabloda prosesin kullandığı çekirdek nesneleri ile alakalı çeşitli bilgiler bulunmaktadır.
- Tablonun detayları microsoft tarafından açıklanmamıştır.
- Fakat incelemeler sonunda bu tablonun temel yapısı hakkında belirli bilgiler edinilmiştir.

Index (handle)	Pointer to Kernel Object Memory Block	Access Mask (DWORD of Flag Bits)	Flags (DWORD of Flag Bits)
1	0x????????	0x????????	0x00000001
2	0x????????	0x????????	0x00000000

Proses ve Thread

- Bir proses oluşturulurken prosesin işlem yapabilmesi için bir adet de thread(iplik) oluşturulur.
- Kodu asıl işleten Thread dir.
- Proses oluşturulduğunda kernel bir adet proses ve bir adet de thread çekirdek nesnesi oluşturur.
- Bu çekirdek nesneleri ile alakalı bilgiler de prosesin handle tablosuna eklenir.
- Bu sayede proses istediğinde kendi çekirdek nesnesi ile alakalı işlemleri çekirdekten talep edebilecektir.

Uygulama Oluşturmak

- Window iki tür uygulama geliştirebilir.
  - Console Uygulaması (CUI)
  - Pencere Uygulaması (GUI)
- Konsol için giriş fonksiyonu

```
int main ( int argc,char *argv[])
```

- Windows için giriş fonksiyonu

```
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR CommandLine,
                   int nShow)
```

CreateProcess

- Proses oluşturmaya yarayan fonksiyondur.
- Direkt olarak Google da taratılırsa fonksiyonun prototipine ulaşmak mümkündür
- İlk Parametre Çalıştırılacak prosesin adını belirtir
- İkinci parametre çalıştırılacak prosese yollanan Command Line verisini yollamak için kullanılır.
- Eğer ilk parametre NULL girilirse otomatik olarak ikinci parametre çalıştırılacak olan program ismi olarak seçilir.
- Oluşturulacak proses için bir proses ve bir de iplik çekirdek nesnesi oluşturulacağından bunların güvenlik ayarları 3. ve 4. parametrede girilir
- Bu parametrelere NULL değerleri girersek sistem varsayılan (default) güvenlik ayarlarını verecektir.
- 5.Parametre prosesin handle tablosunun çocuk prosese aktarılıp aktarılmayacağını belirtir. TRUE girilirse handle tablosu çocuk prosese aktarılır

CreateProcess

- 6.Parametre çalıştırılacak prosesin oluşturulması ile alakalı bayraklar girilir. Örneğin:
  - CREATE_NEW_CONSOLE oluşturulan prosese ait yeni bir console oluşturur verir.
- 7.Parametre oluşturulacak prosesin çevre değişkenlerini belirtir.
- 8.Parametre oluşturulacak prosesin varsayılan klasörünü belirtir.
- 9.Parametre oluşturulacak prosesin başlangıç görüntüsü hakkında bilgi vermemizi sağlar.

CreateProcess

- Son parametre oluşturulan proses hakkında bilgiler ile doldurulur.

```
typedef struct _PROCESS_INFORMATION {
   HANDLE hProcess;
   HANDLE hThread;
   DWORD  dwProcessId;
   DWORD  dwThreadId;
} PROCESS_INFORMATION;
```

- Bu yapıda ilk iki parametre sırası ile oluşturulacak prosesin ve ona ait thread in kernel nesnelerinin handle değerleridir. (Çok uzun bir cümle biliyorum)
- Ebeveyn proses bir çocuk proses oluşturduğunda yukarıda bahsedilen iki çekirdek(kernel) nesnesinin handle (indeks) değerleri ebeveyn prosesin handle tablouna eklenir.
- Son iki parametre ise oluşturulan proses ve thread in ID değeridir.
- Handle değerleri proseslere özgüdür. Aynı kernel nesnesi farklı proseslerde farklı handle değeri alabilir
- Fakat ID değeri sistem içerisinde tektir.Bütün kernel nesnelerinde bu değer sadece Thread ve Proses nesnelerinde bulunmaktadır.
- Bu değeri kullanarak bir prosese her noktadan erişebilir.

CreateProcess

- Fonksiyon TRUE değeri döndürür ise proses oluşturulmuş demektir.