İş parçacıkları (Threads)

Thread – **Ayni adres alanında çalışan hafif proses** *(lightweight process (LWP))* – basic unit of CPU utilization

Paylaşılmayan bilgiler

- Thread'e özel Program Sayacı
- Saklayıcı (yazmaç) seti (Register set)
- Yığın(Stack)

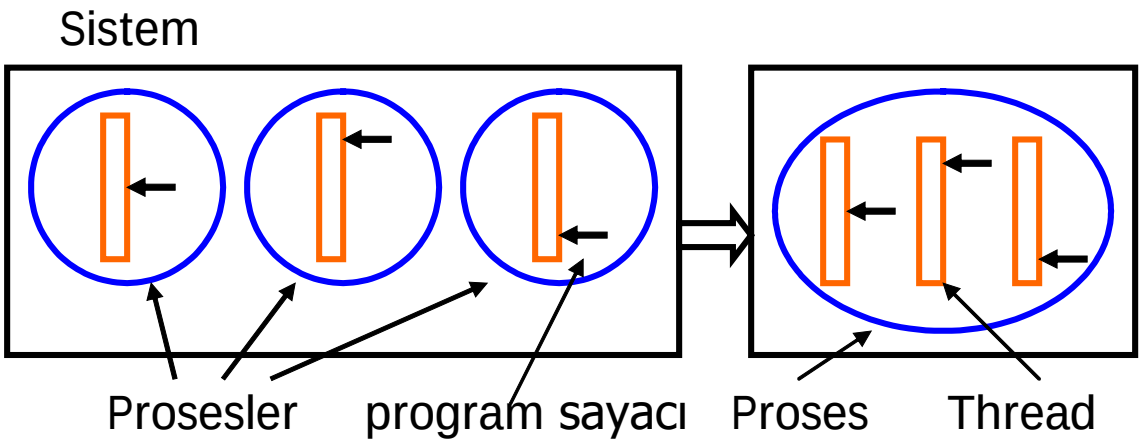
Paylaşılan bilgiler

- Kod
- Veri
- OS kaynakları – açık dosyalar, sinyaller

Ağır proses(Heavyweight process) – geleneksel bir-thread'li iş(task).

Prosesler arası veri paylaşımının zorluklarının çözümünde threadler kullanılır. Ayni proses ortamında birden fazla işlem yürütme imkanı sağlarlar. Thread'ler birbirlerinden bağımsız değillerdir. Thread durumları, çalışır, askıda ve hazır olmak üzere proses durumları ile aynidir. Her prosesin başlangıçta bir thread'I vardır.

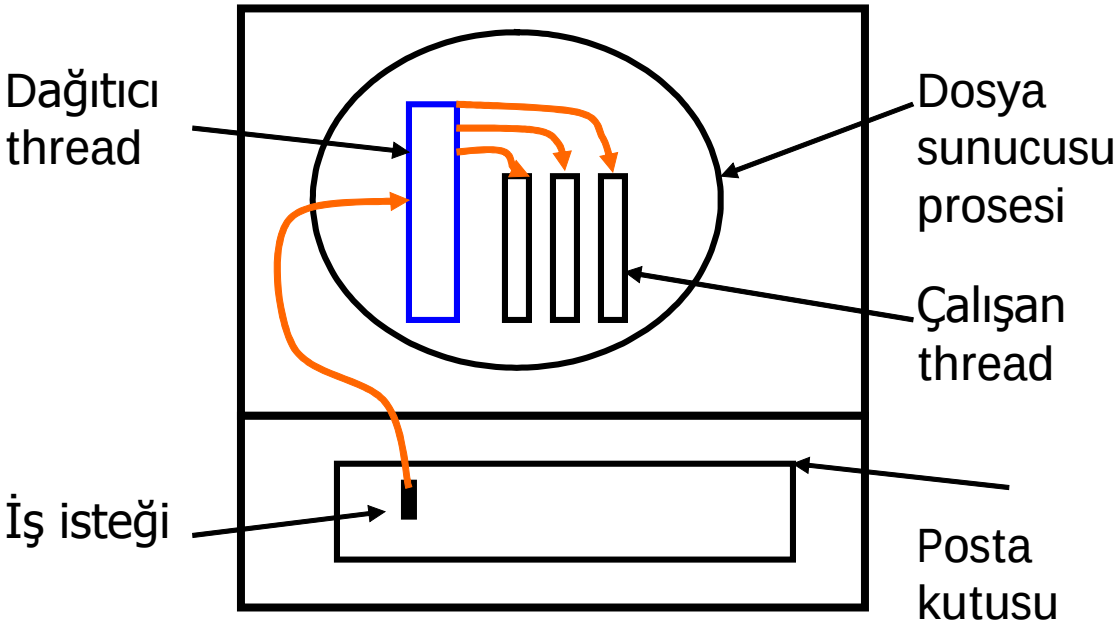
Proses'ten thread'e geçiş



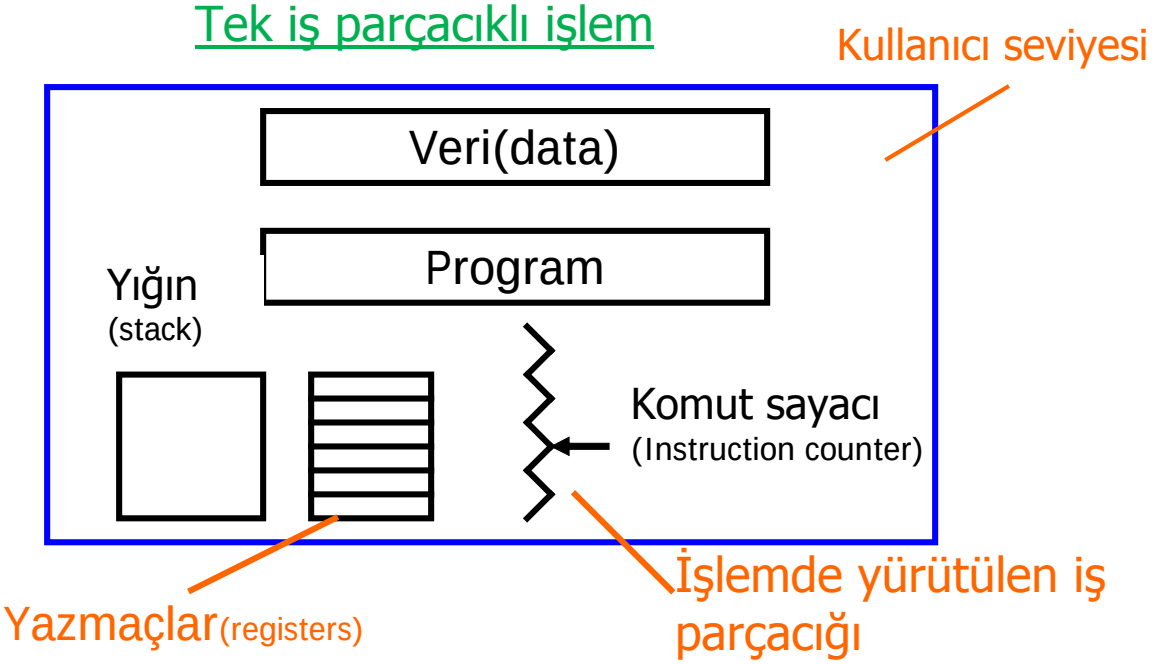
Her proses için	Her Thread için
Adres alanı(Address space) Açılan dosyalar Alt prosesler Semaforlar Sinyaller	Program sayacı Saklayıcılar (Registers) Yığın(Stack) Durum(State) Alt (Child) thread'ler

Farklı bir bellek alanı yok!!!

Bir sunucunun threadler kullanılarak organizasyonu

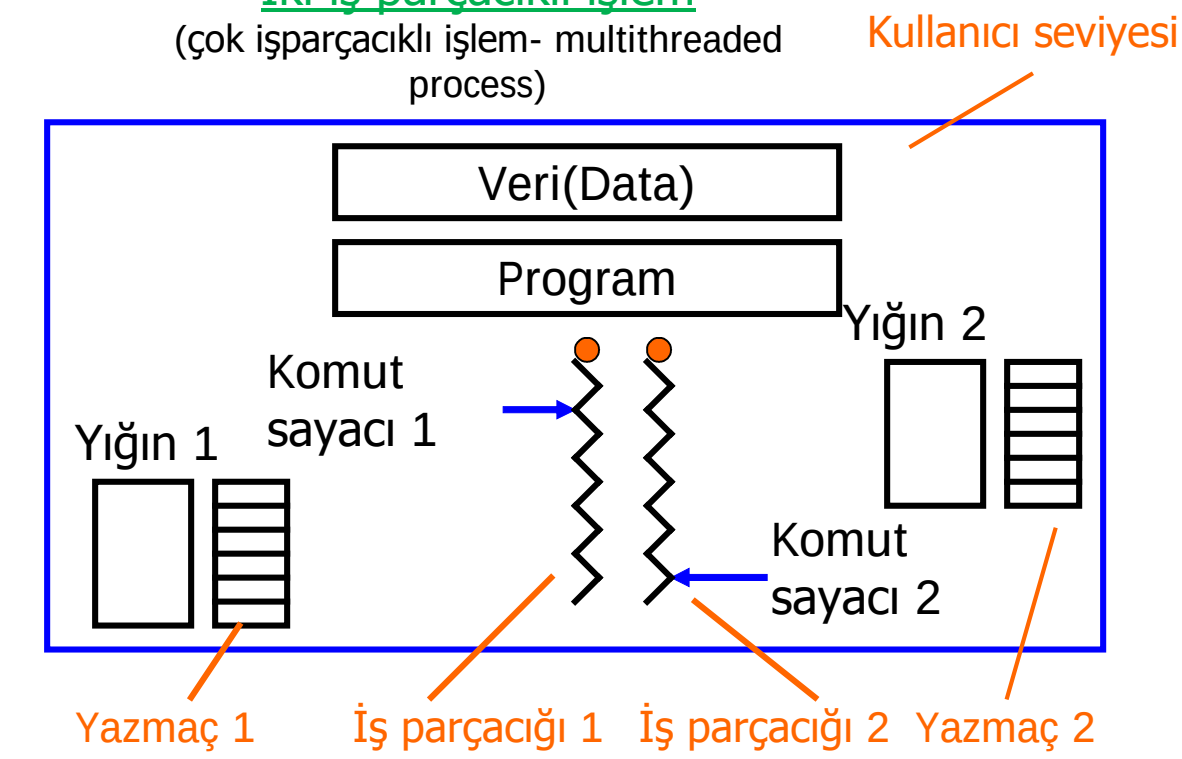


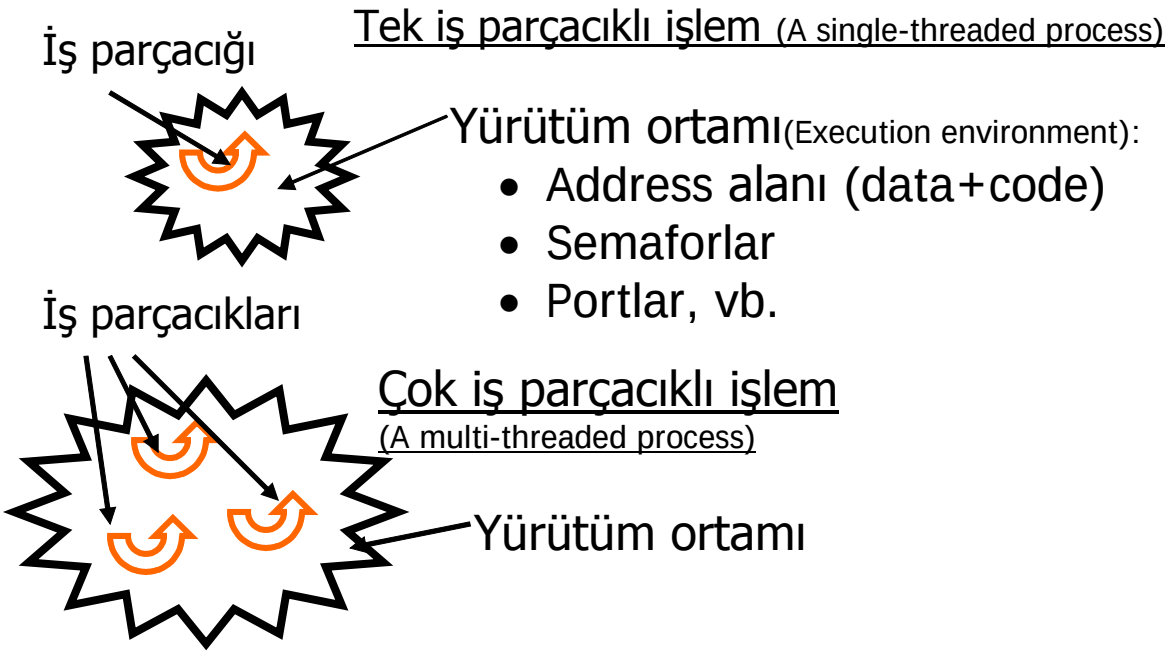
Tek iş parçacıklı işlem



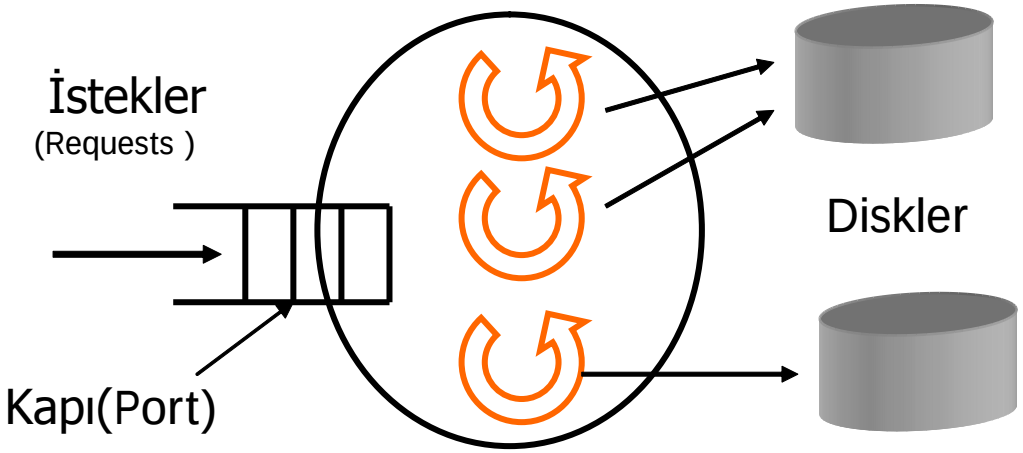
İki iş parçacıklı işlem

(çok işparçacıklı işlem- multithreaded process)





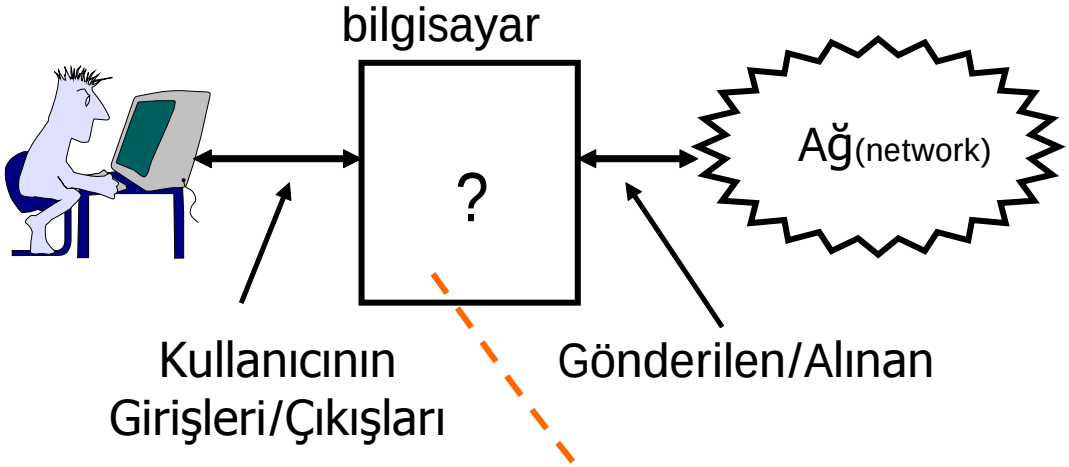
Çok iş parçacıklı sunucuya bir örnek



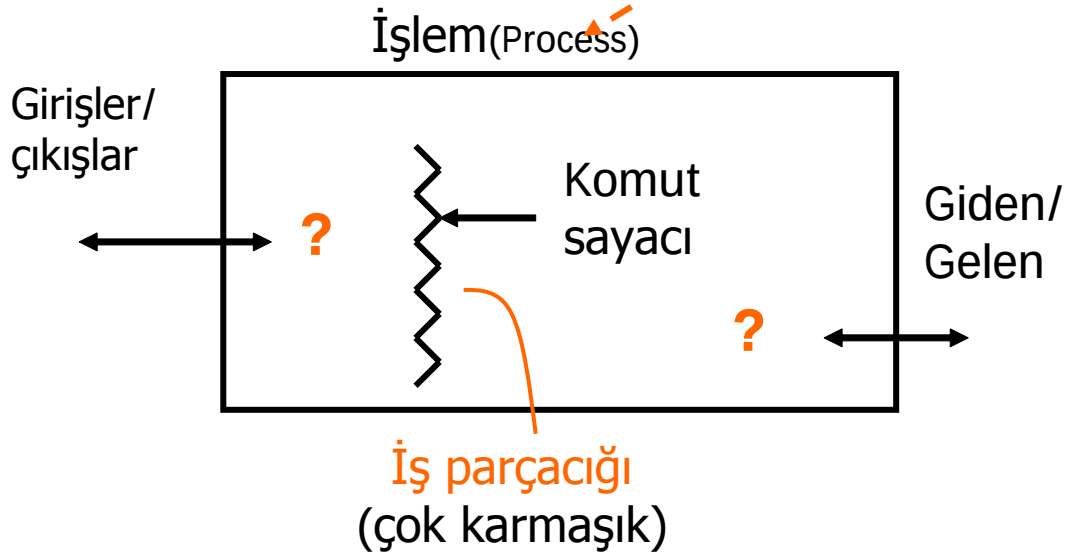
İşlemler ile iş parçacıklarının karşılaştırılması		
Özellik	**Proses**	**Thread**
Yaratma süresi (standart UNIX)	≅ 10 ms	≅ 1 ms
Geçiş süresi	≅ 1.8 ms	≅ 0.4 ms
İletişim mekanizması	Karmaşık	Basit
Veri paylaşımı(Sharing data)	Yok	Var
Koruma(Protection)	Var	Yok
Konum(Location)	Ayni veya farklı bilgisayarlar	Ayni bilgisayar

Thread'ler, işlerin birbirine çok bağlı olduğu,birlikte yürütüldüğü ve paylaşıldığı durumlarda; prosesler ise işlerin birbirinden büyük oranda bağımsız olduğu durumlarda kullanılır.

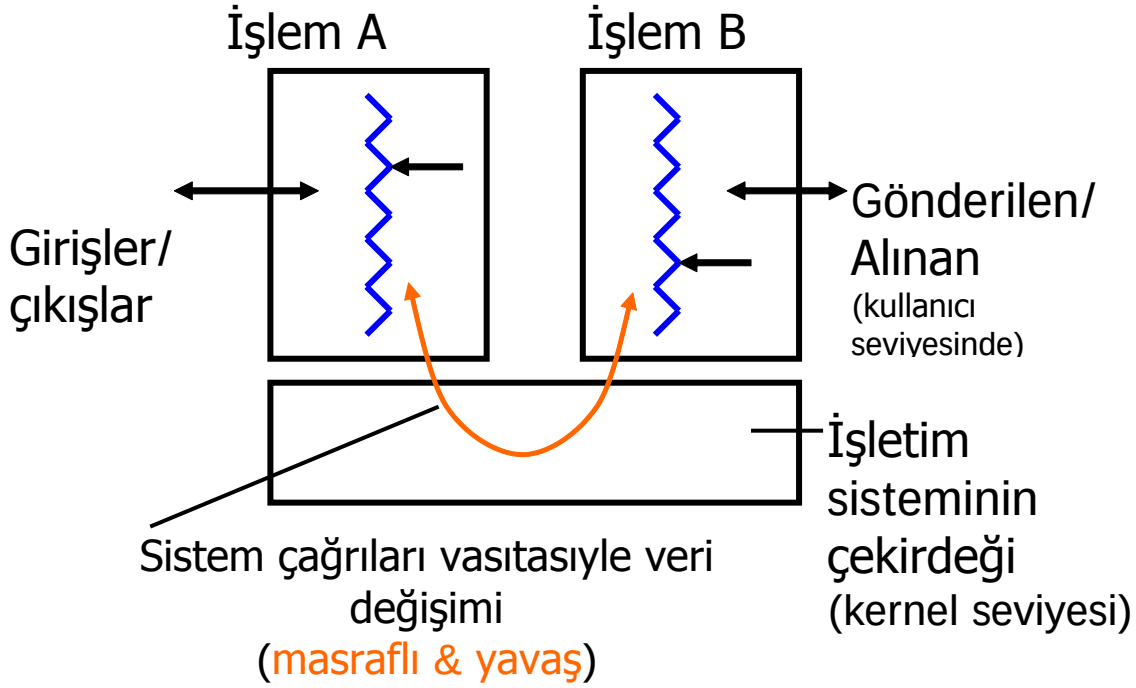
Etkileşimli(interactive) bir sistemin düzenlenmesinde(organization) üç farklı yaklaşım



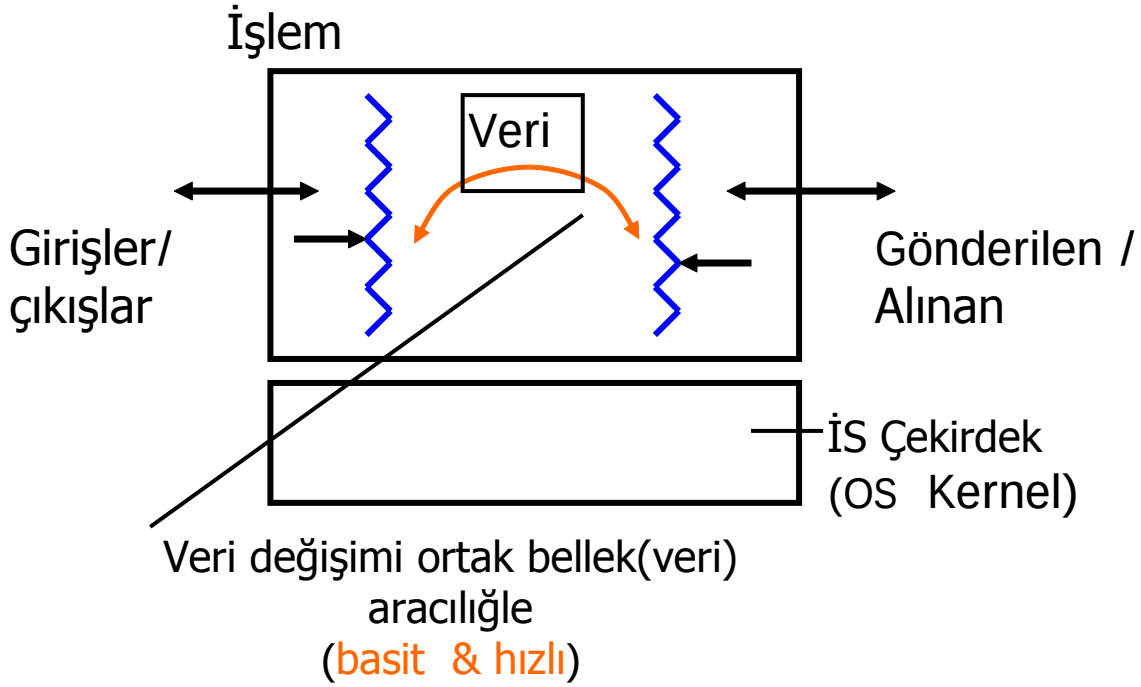
1. Tek iş parçacıklı bir işlem (One process with one thread)



2. İki İşlem (her birinde tek bir iş parçacığı)

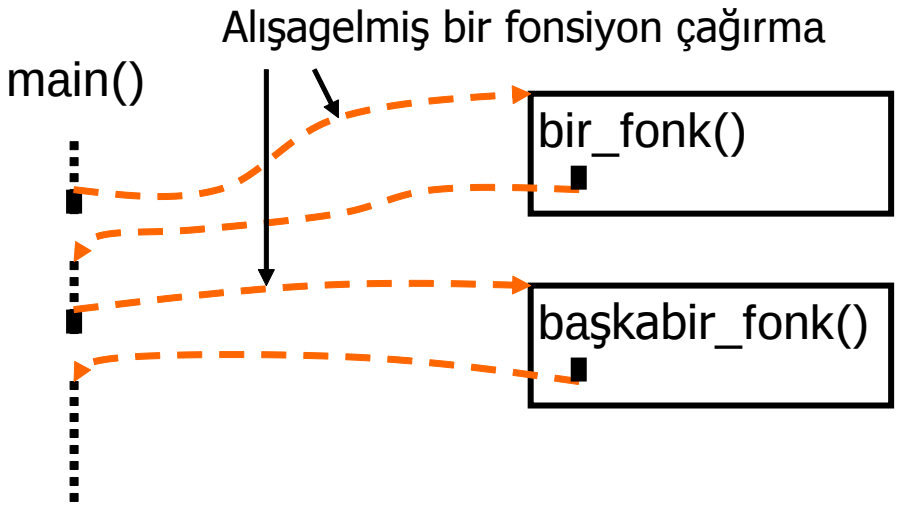


3. İki iş parçacıklı bir işlem(One process with two threads)

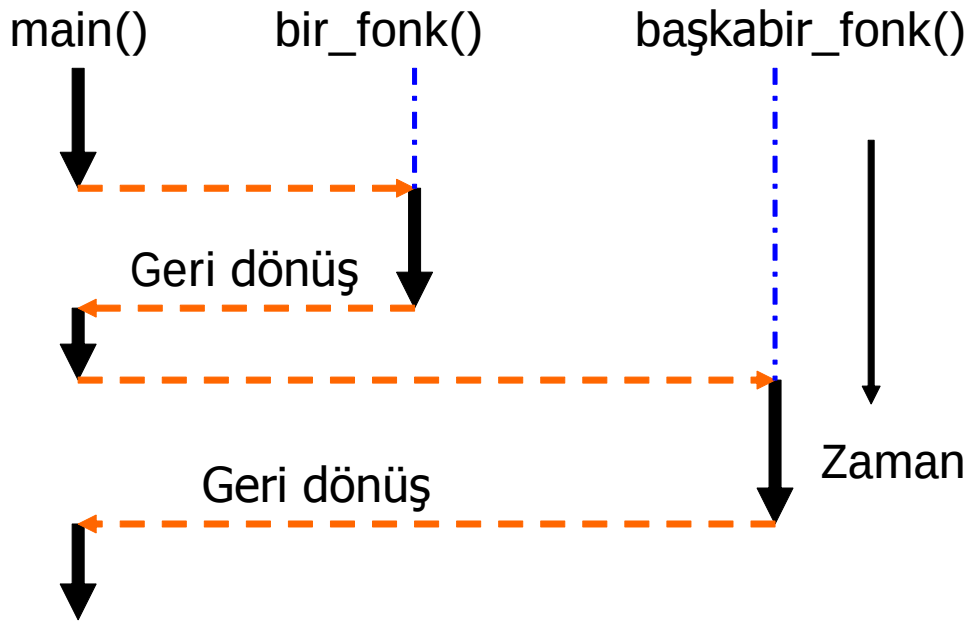


Tek iş parçacıklı bir işlem

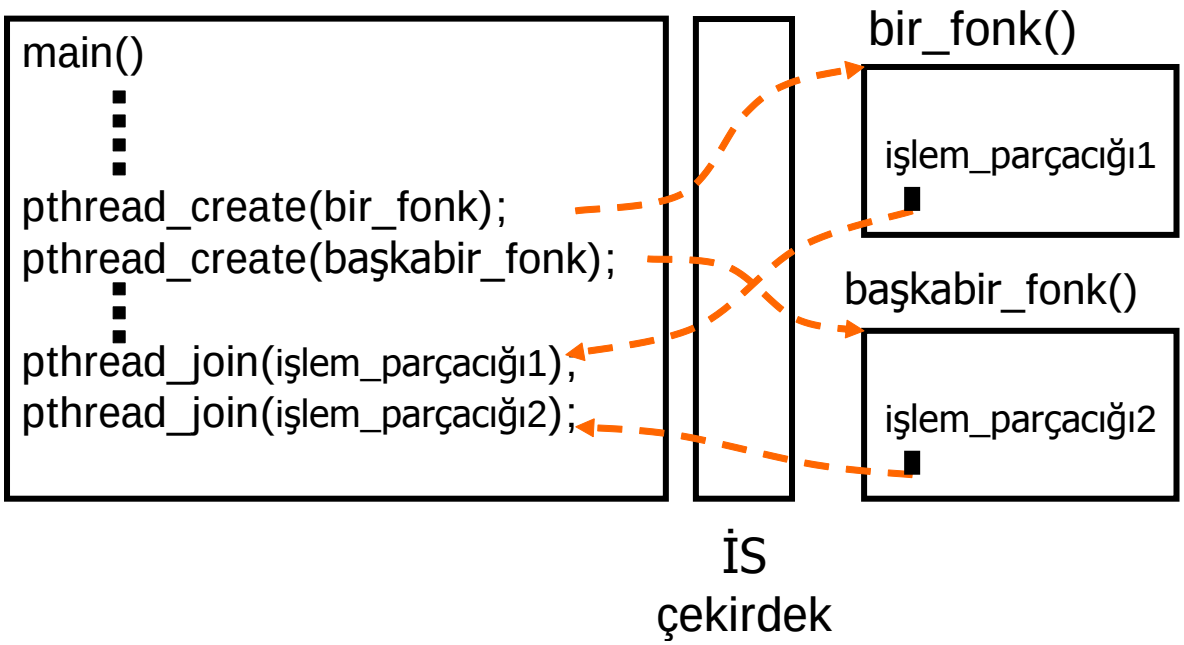
İşlem ardarda iki fonksiyon çağırır(ayni program içerisinde)

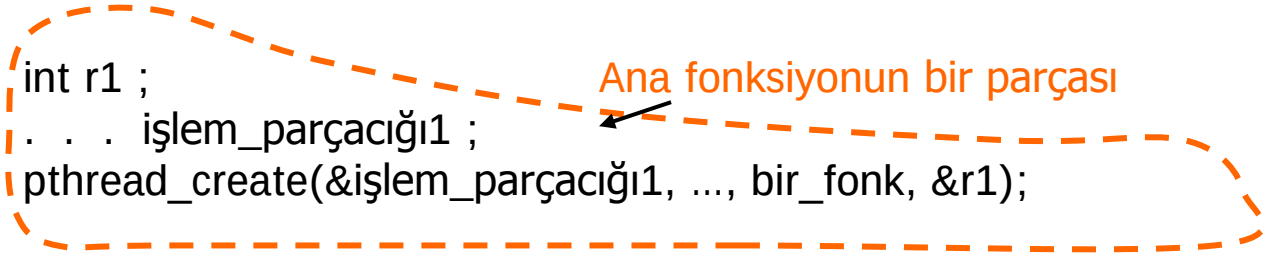


İlgili zaman çizelgesi:

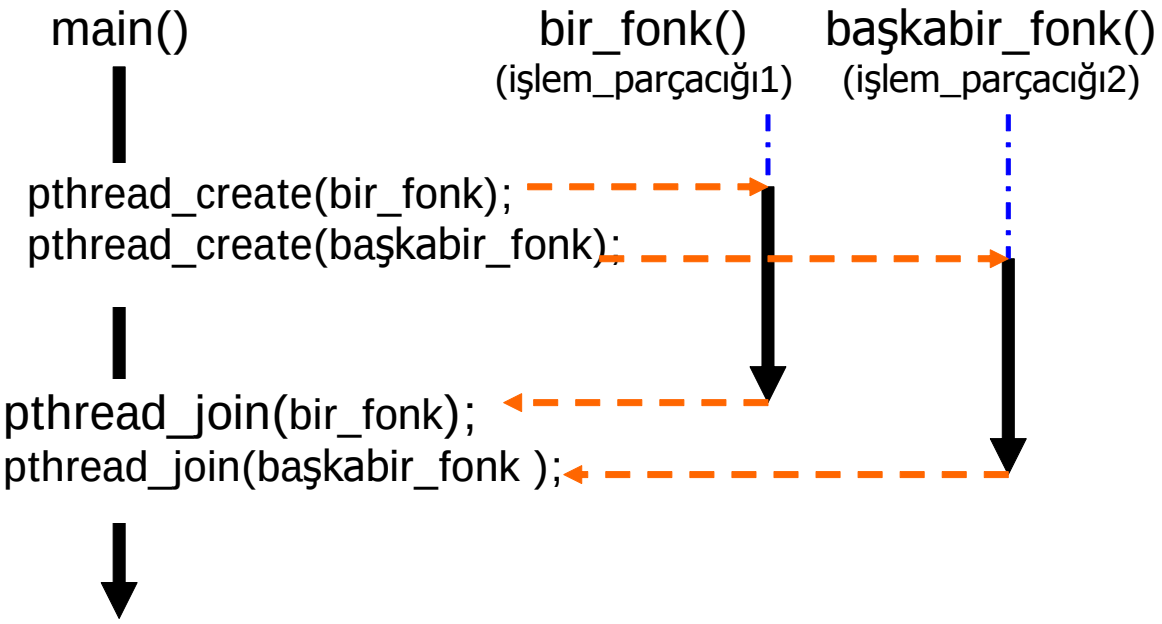


İki işlem parçacıklı bir işlem (Linux işletim sisteminde)





İlgili zaman çizelgesi:



Thread yaratma

Proses yaratılma zamanında *ana (başlangıç)* thread otomatik olarak yaratılır.

Program içerisinde yeni thread'ler yaratmak için:

```
int pthread_create(
      pthread_t          *thread_id,
      pthread_attr_t     *attr,
      void               *(*start_routine)(void),
      void               *arg
 ) ;
```

- Proses içinde bir thread oluşturur ve oluşturulan thread'i fonksiyon ile ilişkilendirir.
- İlişkilendirilen fonksiyon, başka fonksiyon çağırma yapabilir.

- **thread_id** – Yaratılan yeni thread'in kimlik numarası(ID)'nın tutulduğu değişken.
- **attr** – yaratılan threade varsayılan özelliklerin verilmesi için attributes parametresi olarak NULL verilir. Varsayılan özellikler: stack-size, stack-address, scheduling policy, priority, detach state.
- **start_routine** – Çalıştırılacak fonksiyonun adı (sadece bır parameter alabilir – ***arg)***.
- **arg** – Fonksiyona aktarılacak parametre adresi veya veri yapısının işaretleyicisi(pointer to a data structure), -eğer birden fazla parametre gerekliyse. **NULL** – fonksiyon parametre almıyorsa.

Bir thread'in sonlanmasını bekleme

Sonlandırılmış thread'in kaynaklarının iadesi:

```
int pthread_join(
      pthread_t      thread_id,
      void           **return_val
  ) ;
```

Kimlik numarası ***thread_id*** olan thread sonlana kadar çağıran thread'i askıya alır.

- Eğer *thread_id* sonlandıysa – hemen döner.
- Proseslerde olduğu gibi ana-alt (Parent-child) ilişkisi yoktur – **threadler eştirler** (herhangi bir birleşebilir thread için bekleyebilir).
- Eğer dönen değer ***return_val*** **!= NULL**, ise *thread_id*' nin çıkış durumu (status) *return_val* 'da tutulur.
- *pthread_join()* bütün yaratılan threadler için çağrılmalıdır.

Proses Sonlandırma

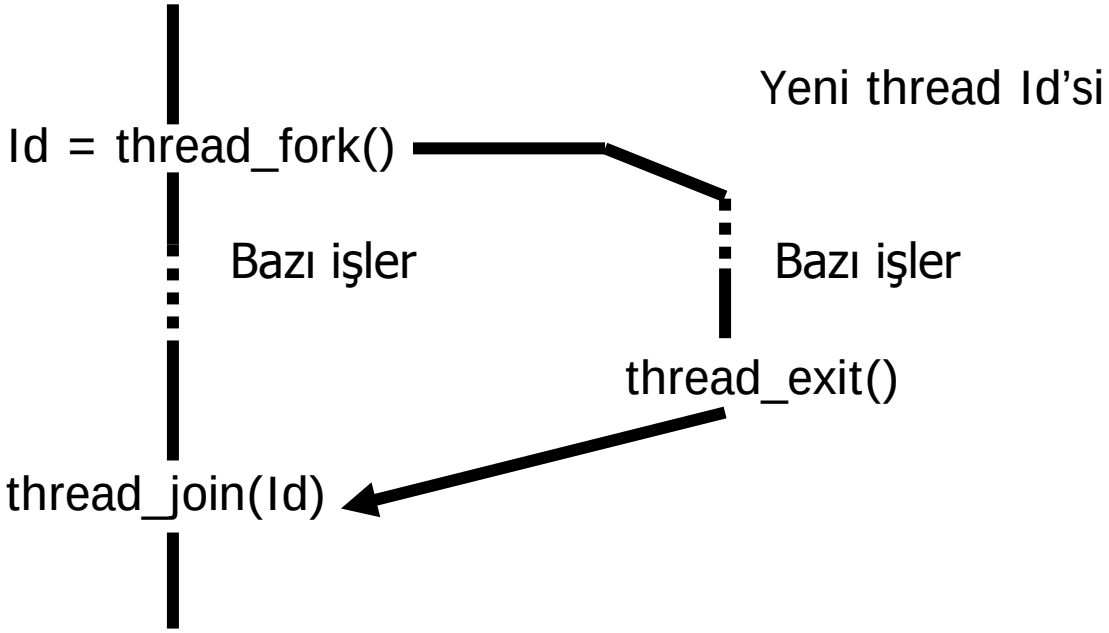
Eğer bir proses *exit() _exit()*, kullanırsa, once, proses içerisindeki bütün thread'ler sonlandırılır, sonra da proses sonlandırılır.

Pencere(Windows) sitemleri için bir işlem parçacıklı program

```
"include" & "define" directives
Değişken tanımlamaları

unsigned _stdcall Threadname(argümanlar)
{
İş parçacığı tarafından yapılan iş
return 0 ;
}

main()
. . . .
{
. . . .
hThread= _beginthreadex(..., İşparçacığı_ismi, argümanlar) ;

    Ana fonksiyon işini yapar . . .

WaitForSingleObject(hThread, ... ) ;
CloseHandle(hThread);
. . . .
return 0 ;
}
```

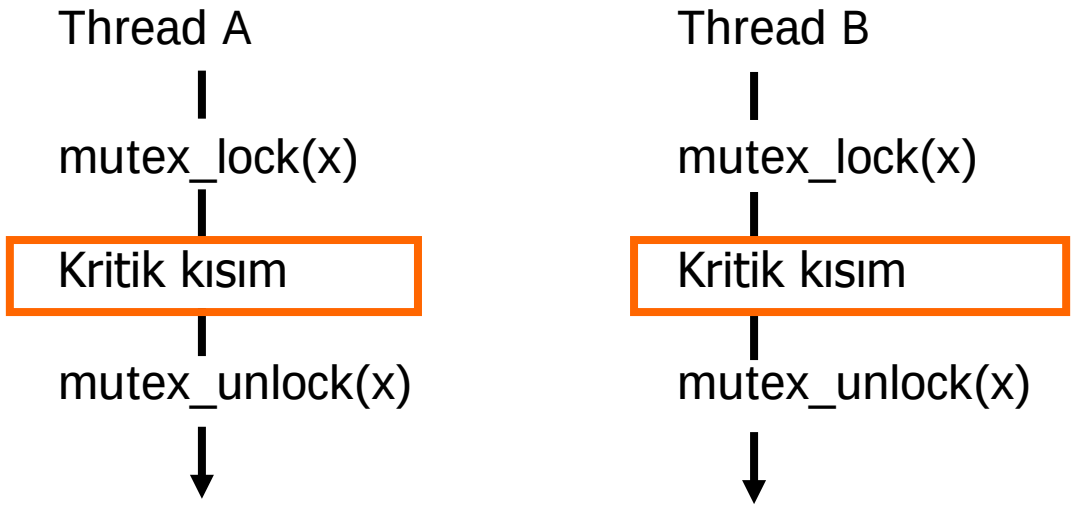
İş parçacığı fonksiyon

Temel thread çağrıları (Mach OS için)



Thread Eşzamanlama (synchronization) için sistem çağrıları

mutex_lock(mutexId)
mutex_unlock(mutexId)
condition_wait(conditionId, mutexId)
condition_signal(conditionId)



UNİX için iki iş parçacıklı(ana iş parçacığı ve yaratılan iş parçacığı) bir program

```
#include<stdlib.h>
#include<stdio.h>
#include<pthread.h>
/* Burda tanımlanan herhangi bir veri her iki iş parçacığı
tarafından kullanılabilir*/

void fonk1(void);  /* Fonksiyonun tanımı */

int main(void)
{
   pthread_t td1;
   int p, j;
   printf("İşlem ana iş parçacığı olarak başlar...\n");
/* Yeni işlem parçacığı yaratılır*/
   p = pthread_create ( &td1, NULL, fonk1, NULL);
   if (p != 0) { perror ("Thread problemi"); exit(1); }
   for(j=1; j<=4; ++j) printf("Ana iş parçacığı  çalışır:%d\n", j);
   pthread_join(td1, NULL); /*yaratılan iş parçacığı için bekler*/
   printf("Ana iş parçacığı sonlanır...\n");
   exit(0);
}
void fonk1(void)
{  int i;
    for(i = 1; i <= 3; i++)
    printf("yaratılan iş parçacığı:%d\n", i);
return;
}
```

opsiyon

```
%gcc –o threads threads.c  –lpthread
```

-lpthread: thread kütüphanesi

```
<9:14:01>~/threads%thread1
İşlem ana iş parçacığı olarak
başlar...
Ana iş parçacığı çalışır: 1
Ana iş parçacığı çalışır: 2
Ana iş parçacığı çalışır: 3
Ana iş parçacığı çalışır: 4
Yaratılan iş parçacığı :1
Yaratılan iş parçacığı:2
Yaratılan iş parçacığı :3
Ana iş parçacığı sonlanır...
<9:14:04>~/threads%
```

Beş iş parçacıklı(ana iş parçacığı ve 4 tane yeni yaratılan iş parçacığı) bir program

```
#include<stdlib.h>
#include<stdio.h>
#include<pthread.h>
/* Burda tanımlanan herhangi bir veri tüm iş
parçacıkları tarafından kullanılabilir*/

void func1(int *i)

int main(void)
{
   pthread_t tids[4];
   int i, p, vals[4];
   for (i=0; i<4; i++)
   {
     vals[i] = i;
      p = pthread_create(tids+i, NULL, fonk1, vals+i);
     if(p!=0)
     {
        perror("Thread problem");
        exit(1);
     }
   }
   for(i=0; i<4; i++)
   {
    printf("İş parçacığı numarası %d ile birleşmeye
    çalışıyor…\n", i);
     pthread_join(tids[i], NULL);
    printf("İş parçacığı numarası %d ile birleşti\n",
    i);
   }
   exit(0);
}

void fonk1(int *i)
{
  printf("Merhaba.
         Ben iş parçacığı %d\n", *i);
  return;
}
```

Bir çıktı:

```
<9:40:45>~/threads%thread2

Merhaba. Ben iş parçacığı 0
Merhaba. Ben iş parçacığı 1
Merhaba. Ben iş parçacığı 2
iş parçacığı numarası 0 ile birleşmeye çalışıyor…
İş parçacığı numarası 0 ile birleşti
iş parçacığı numarası 1 ile birleşmeye çalışıyor…
İş parçacığı numarası 1 ile birleşti
iş parçacığı numarası 2 ile birleşmeye çalışıyor…
İş parçacığı numarası 2 ile birleşti
iş parçacığı numarası 3 ile birleşmeye çalışıyor…
Merhaba. Ben iş parçacığı 3
İş parçacığı numarası 3 ile birleşti

<9:40:48>~/threads%
```