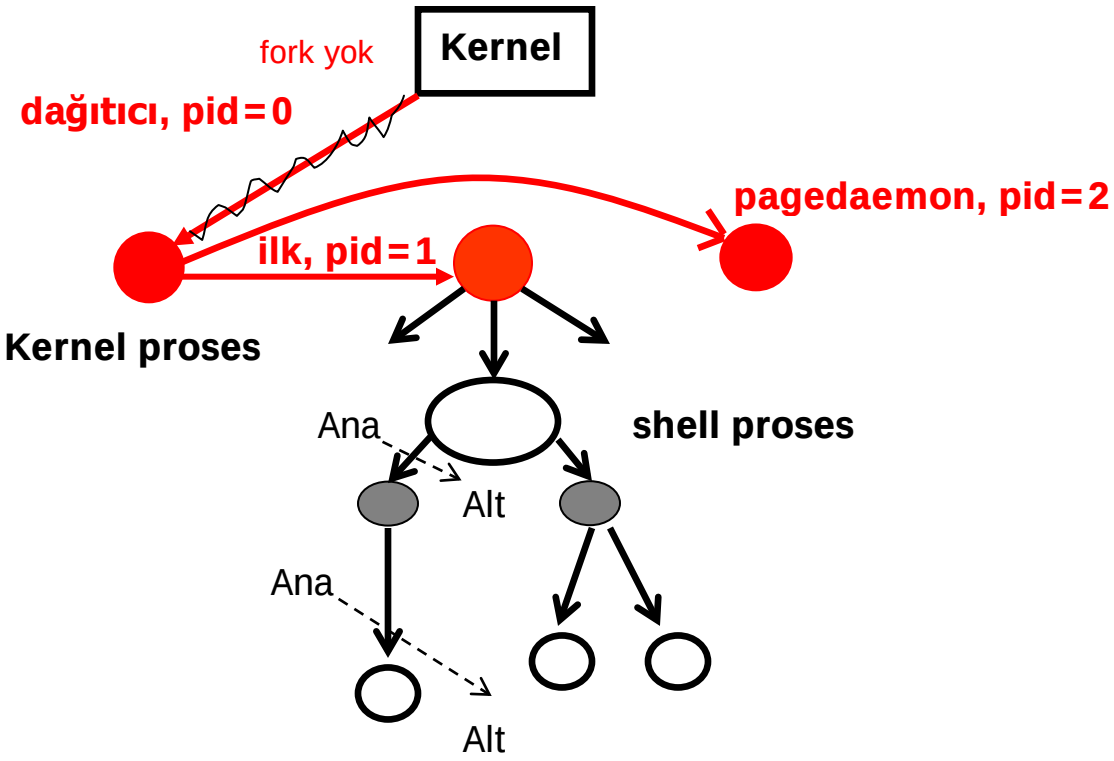
UNIX de İşlem Hiyerarşisi



Ana/Alt işlem ilişkisi

Ana işlem ilk(1) bütün UNIX işlemlerini yaratır (0 ve 2 hariç).

fork, işlem oluşturmak için kullanılan sistem çağrısıdır

```
pid_t   fork ( void ) ;
```

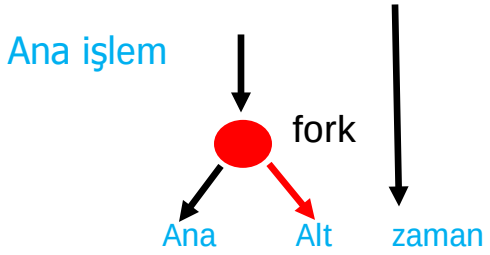
fork() işlemi başarılı olduğunda dönen değer:

- Alt işlemde 0.
- Ana işlemde alt işlemin kimliği (PID).

Eğer başarılı değilse dönen değer, -1 dir

Olası hata mesajları:

- Kaynak geçici olarak kullanılamıyor.
- Yeterli alan yok.



İşlem oluşturabilmek için kısa bir program örneği

```
pid_t pid;
.....
.....
pid=fork();
```

/* Ana işlem talimatları */

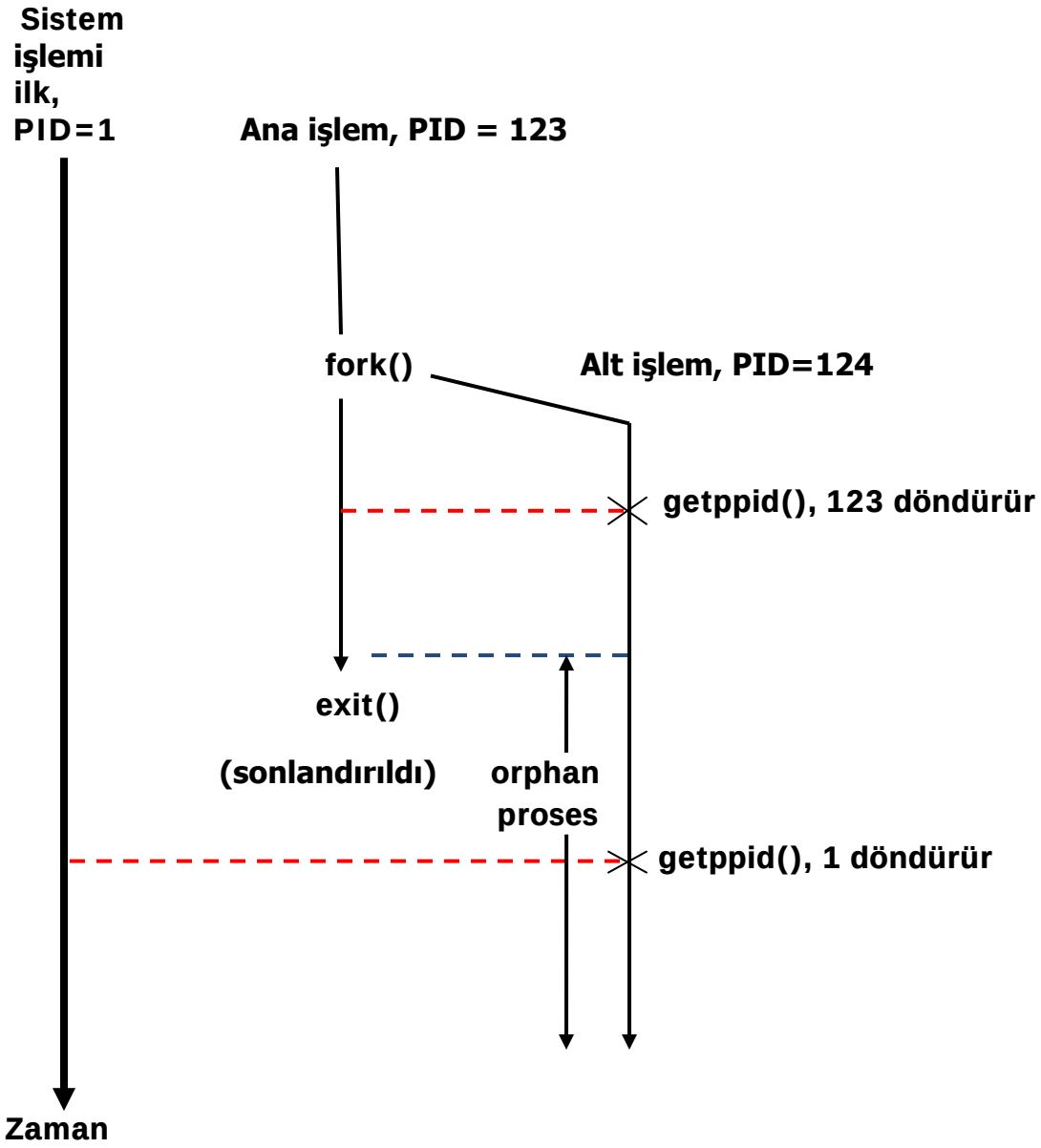
```
if (pid == -1) { perror("fork") ;  exit(1) ; }
if(pid==0)
{. . . . ./* Alt işlem için talimatlar */
. . . . .
}
else
{ . . . ./* Ana işlem için kalan talimatlar */
  . . . .
}
```

Basit bir örnek:

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
main()
{  fork(); printf("Mesaj A\n");
   fork(); printf("Mesaj B\n");
  . . .
}
```

1. Kaç tane işlem çalışacak?
2. Kaç tane mesaj görüntülenecek?

Ana işlem kendisinin alt işleminden önce sonlanırsa durum ne olur?



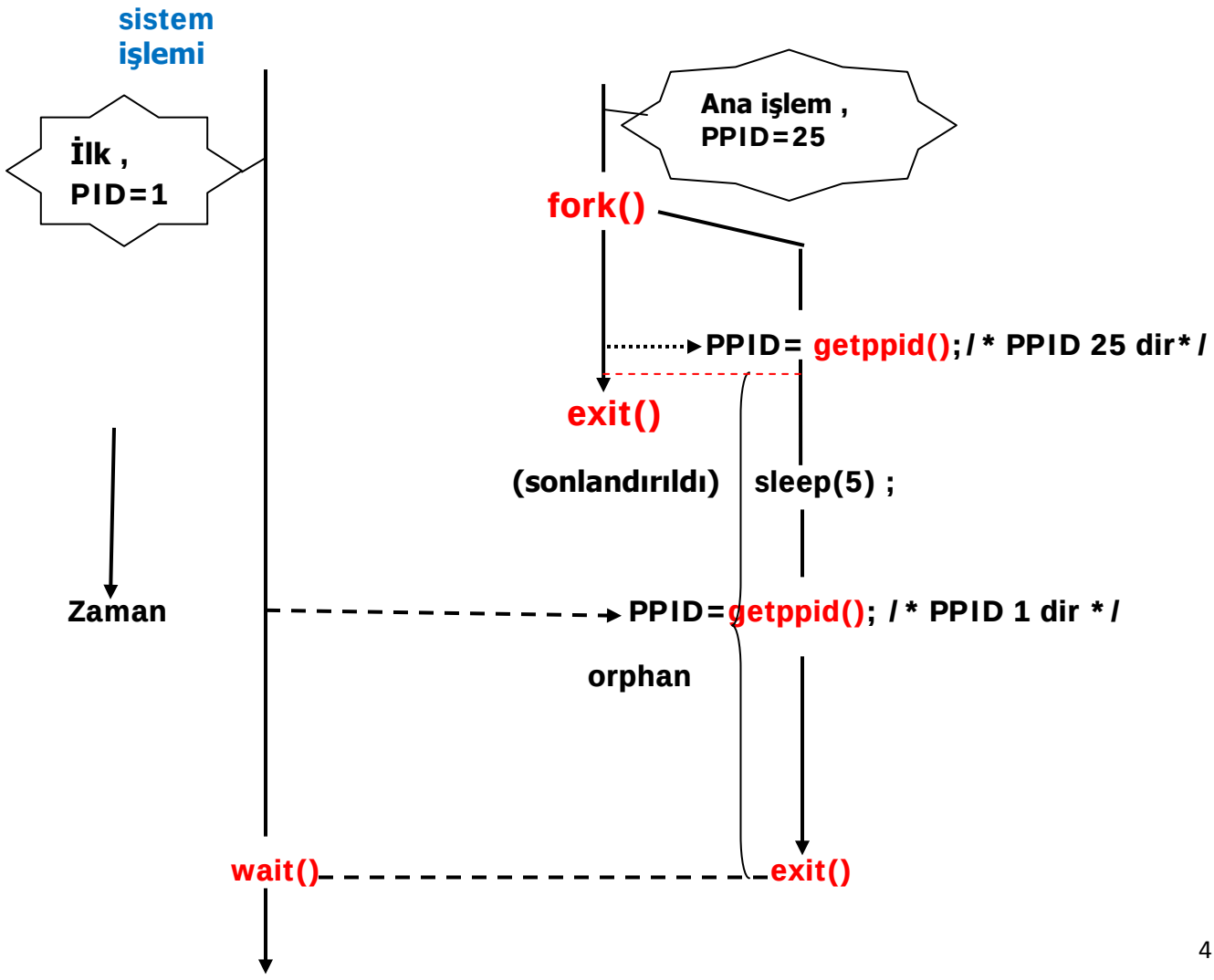
Bir işlem tarafından işlem kimliği (pid) alma

```
      pid_t   getpid(void);
veya int                          <sys/types.h>
      pid_t   getppid(void);      <unistd.h>
```

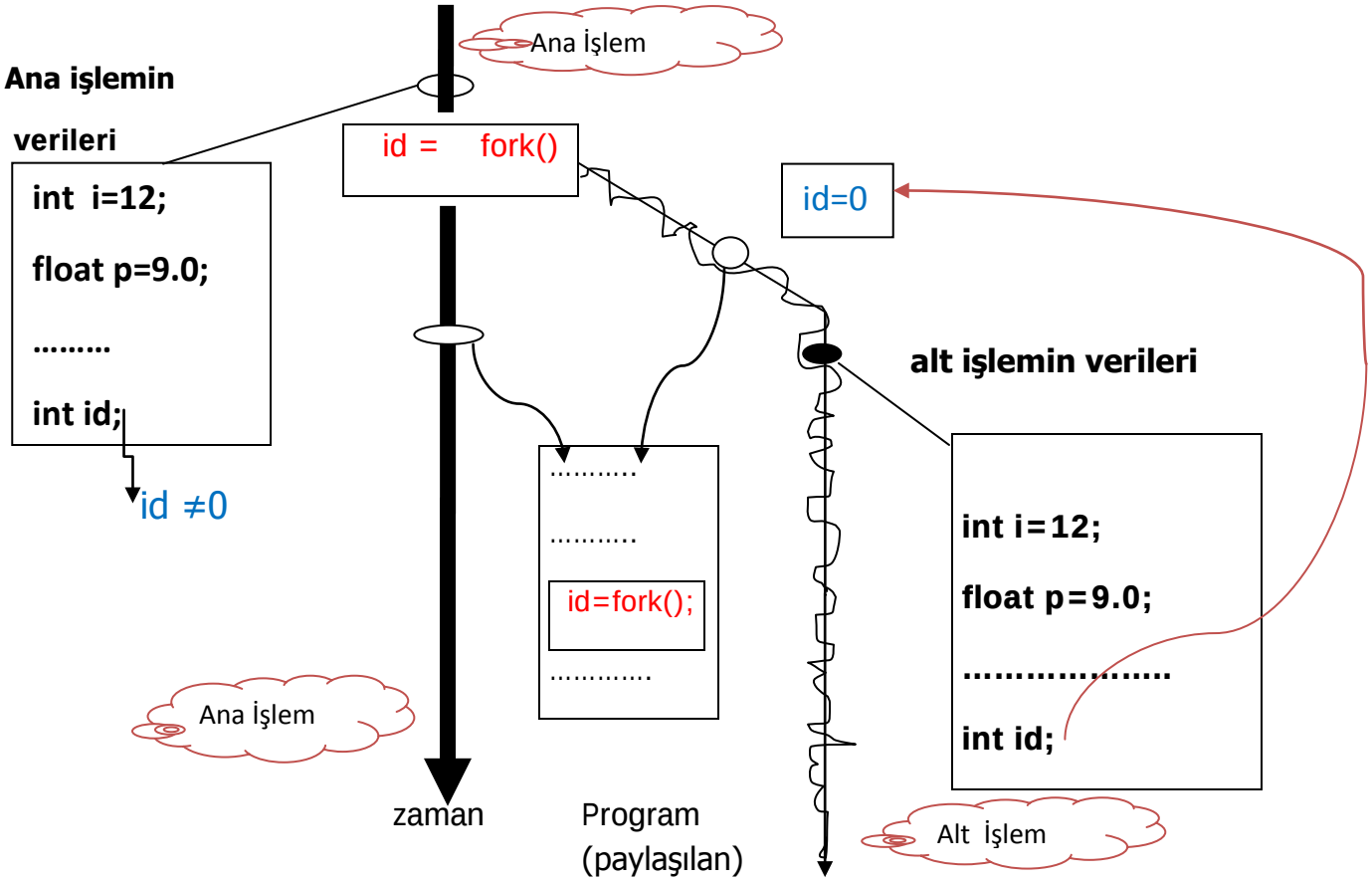
içerdiği dosyalar:

Örnek:

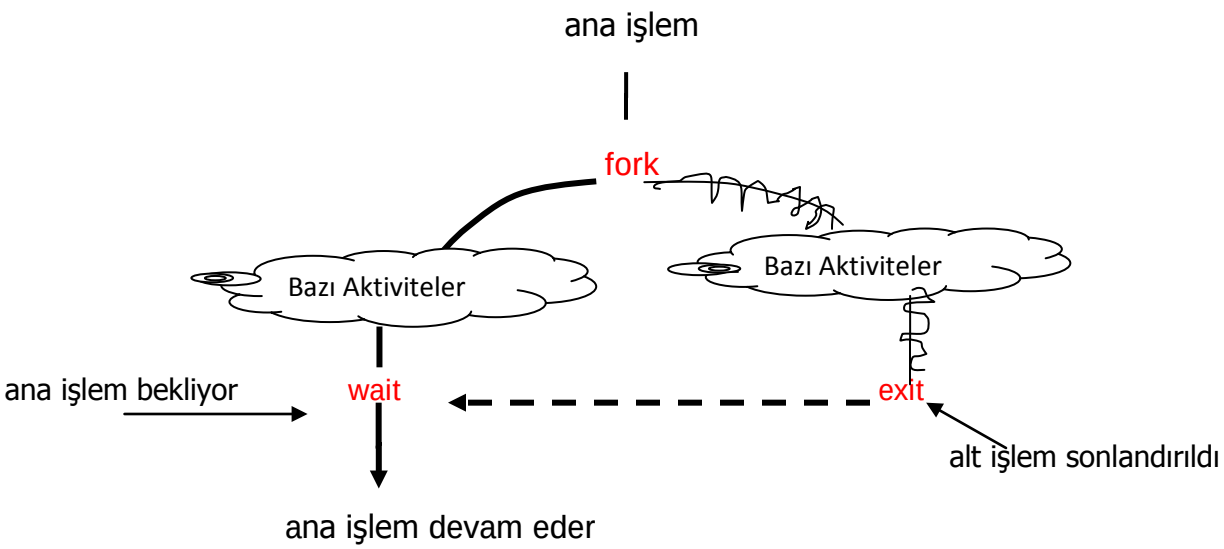
```
printf("Benim işlem kimliğim %d dir.\n",getpid());
printf("Benim ana işlemimim kimliği %d dir.\n",getppid());
```



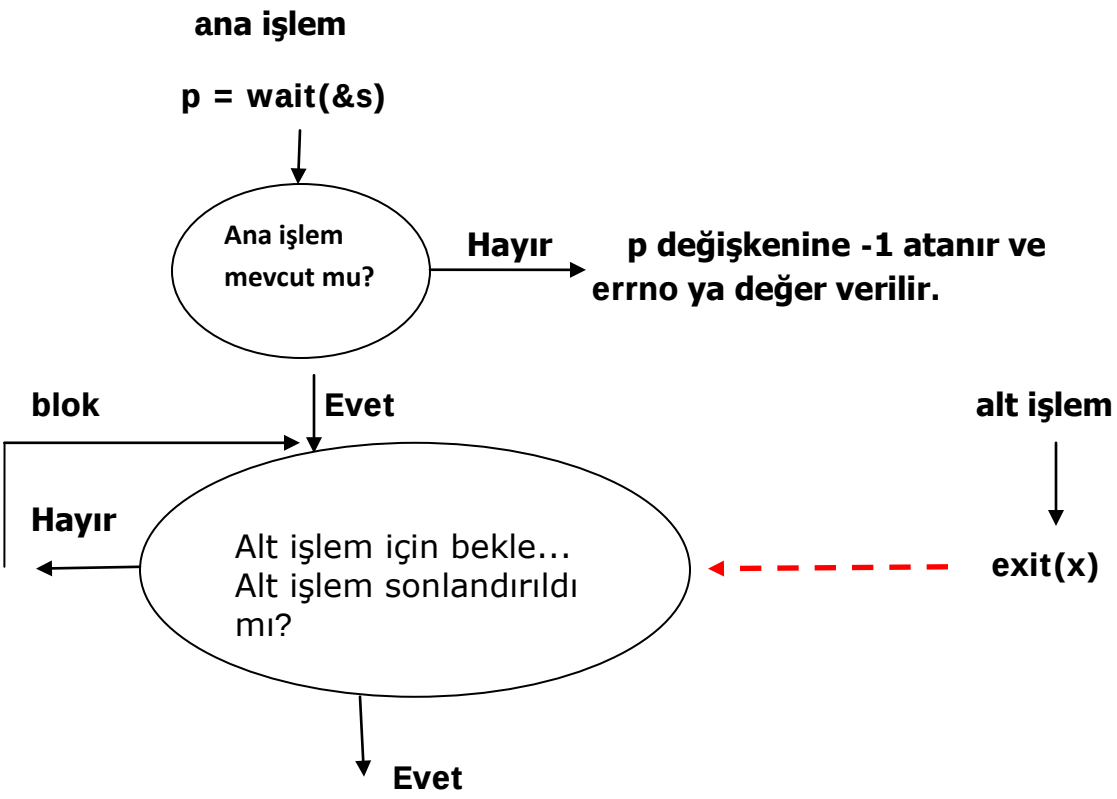
Ana ve alt işlemlerin adres alanları



Ana ve alt işlemler arasında senkronizasyon



UNIX de wait() ve exit() system çağrıları arasındaki ilişki



Alt işlemin kimliği (PID si) P değişkenine atanır,

ve alt işlemin çıkış değeri olan x de s değişkenine atanır.

Eğer wait()` in parametresi 0 yada boş( NULL) ise, wait() sistem çağrısı sadece alt işlemin kimliğini (PID`sini) p değişkenine atar.

Örnek: Bir ana işlem, bütün alt işlemlerin sonlanmasını bekliyor…

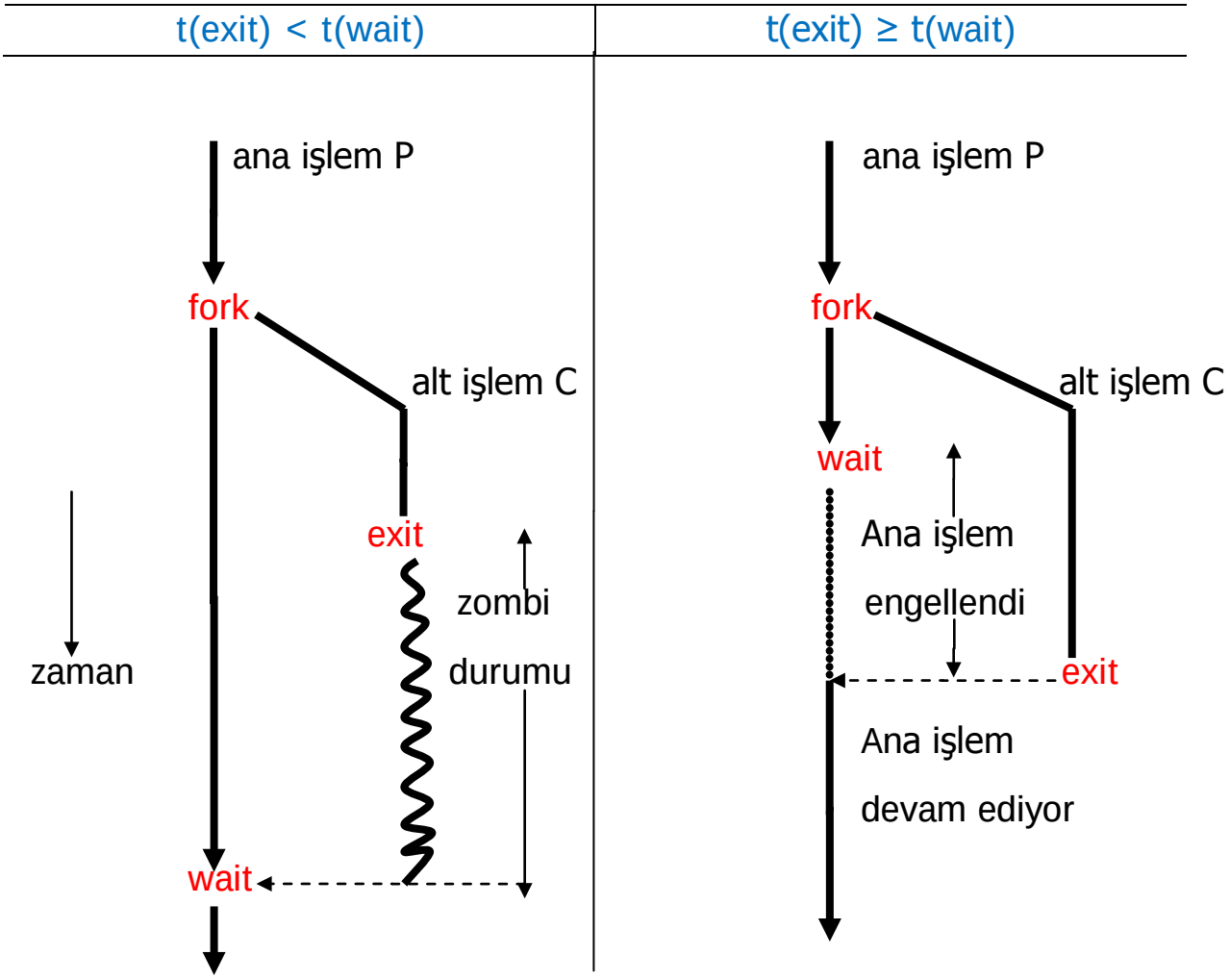
```
int i, pid, status, w;
.
.
.
for(i=0; i<3; i++) /*3 alt işlem başlıyor*/
{
    pid = fork();
    if (id==0){/*alt işlem çalışır ve sonlanır*/}
}
while (( w= wait (&status)) && w != -1)
      printf("Alt işlem PID si: %d,  çıkış kodu: %d\n", w, status);
```

Bir işlemin zombi olma durumu

P'yi bir ana işlem, C'yi bir alt işlem olarak düşünelim.

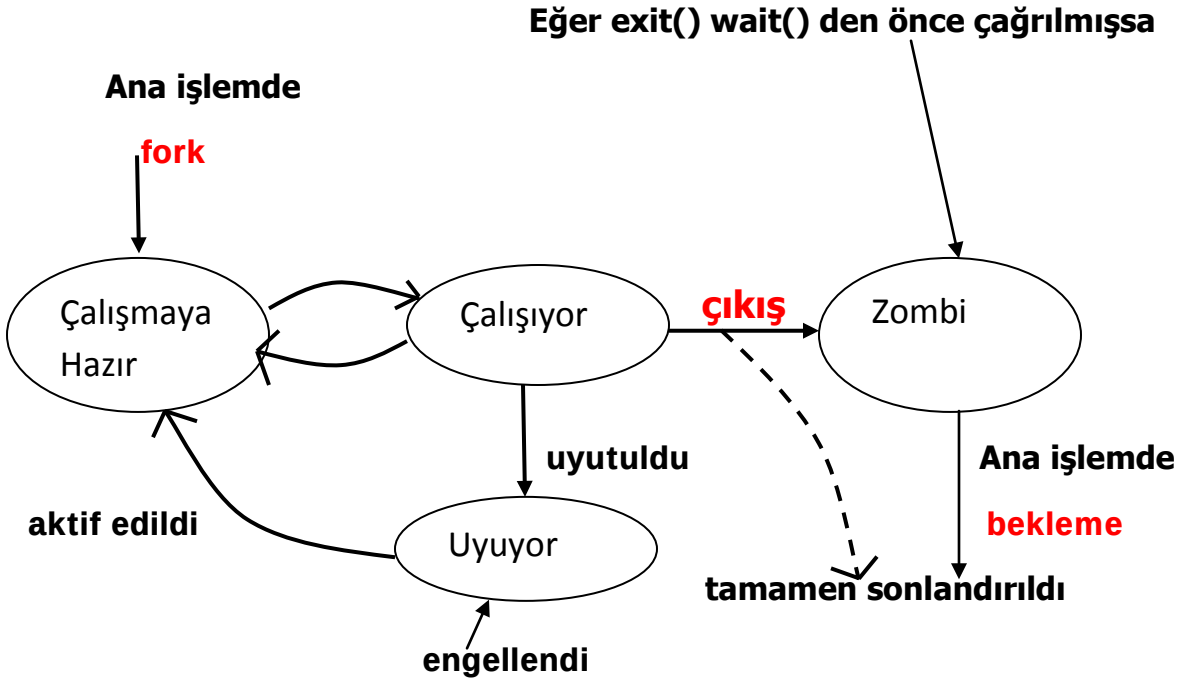
t(fork): P işleminin fork sistem çağrısını çağırdığı zaman
t(exit): C alt işleminin sonlandırıldığı zaman
t(wait): P ana işleminin wait sistem çağrısını çağırdığı zaman

Açıkca , t(fork) < t(exit) , t(fork) < t(wait)



UNIX' de bir işlemin durum diyagramı

(basitleştirilmiş)



İşlemler için bkz. Curry, D., Ch. 11, pp. 283-287, 290-302