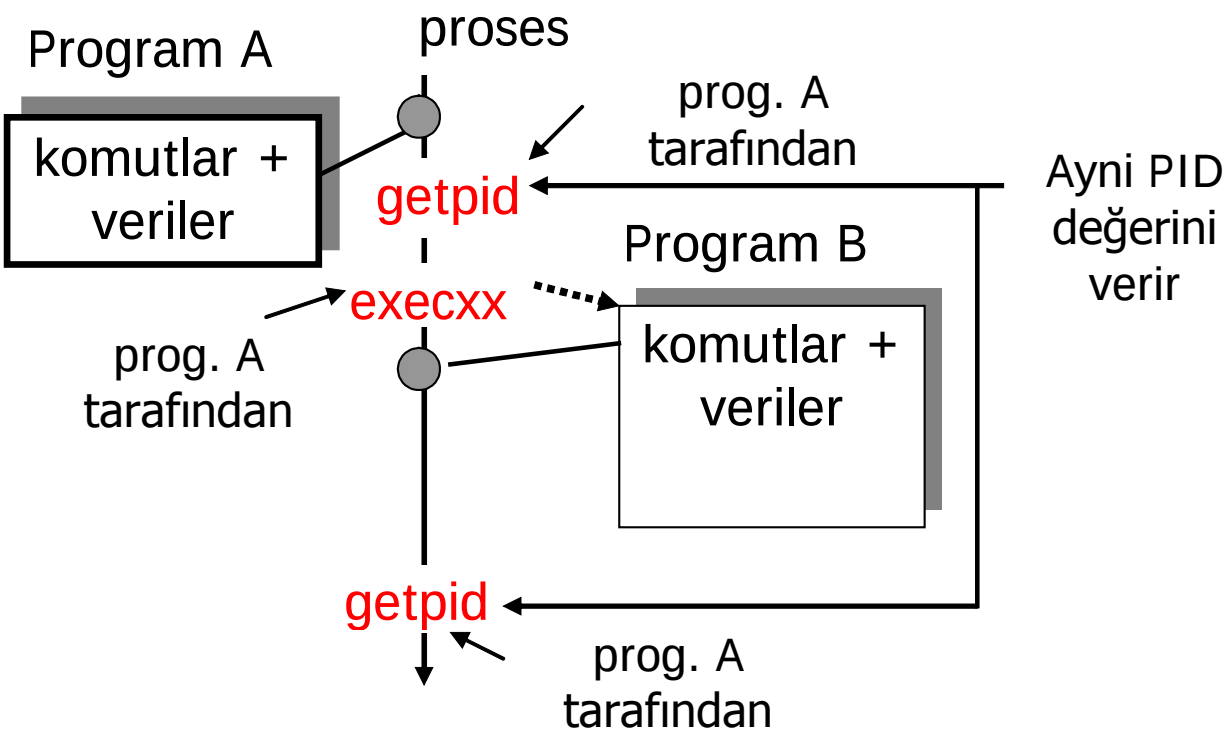
Proses içinde başka bir programa aktarma



#include <unistd.h>

*int execl (const char *path, const char *arg0, ..., const char *argn, char * /*NULL*/) ;*

Başarı durumunda geri dönüş: Yok (geri dönmez)

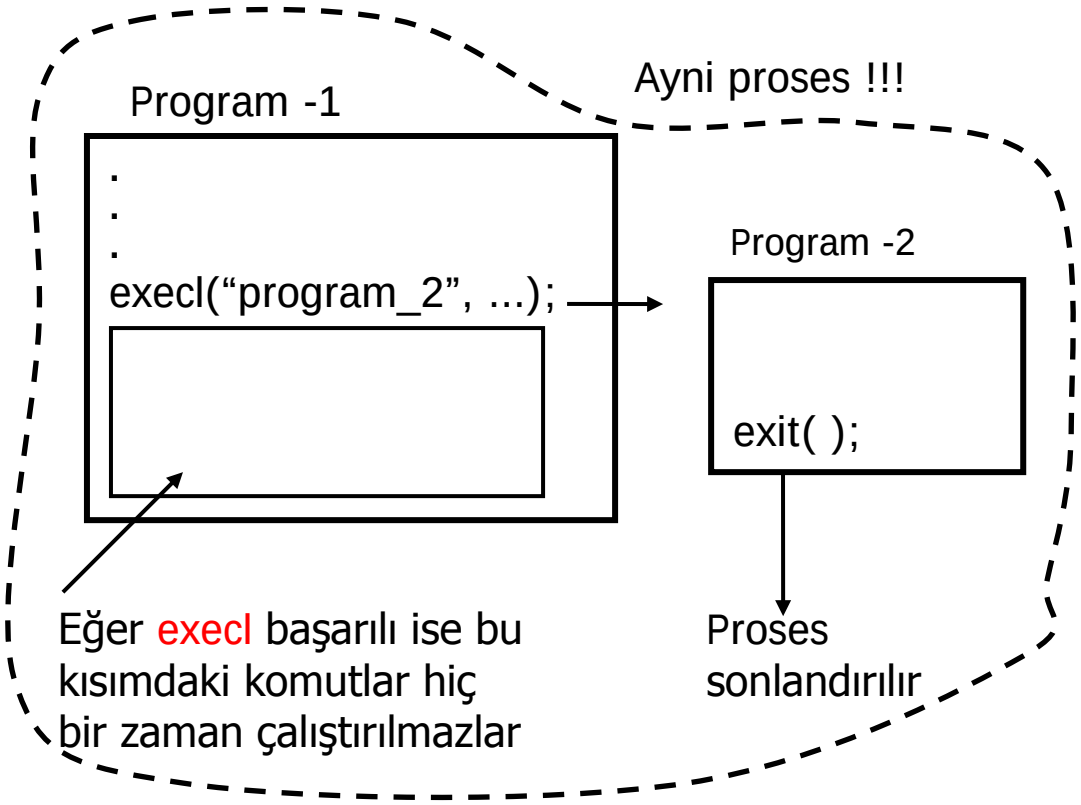
Örnek:

yol arg0 NULL

execl("yeni_program", "yeni_program", 0) ;

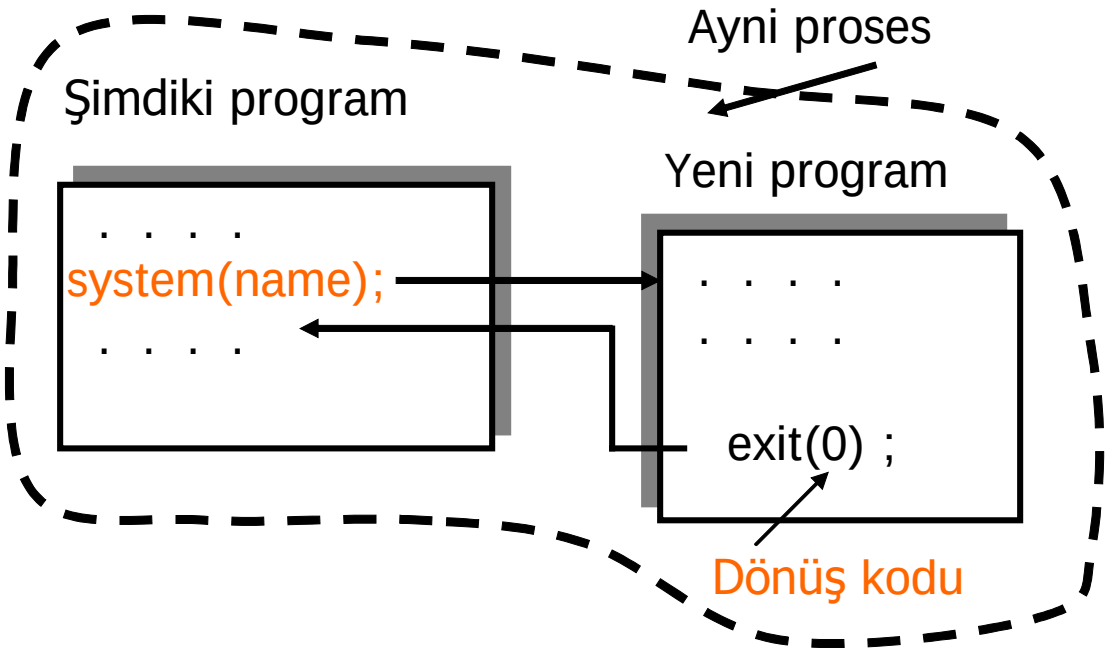
Parametrelerin sonu

Başka bir programa aktarım
(eski programa dönüş olmaksızın)



Çalışan program içerisinden yeni bir programa aktarım(eski programa geri dönüş var)

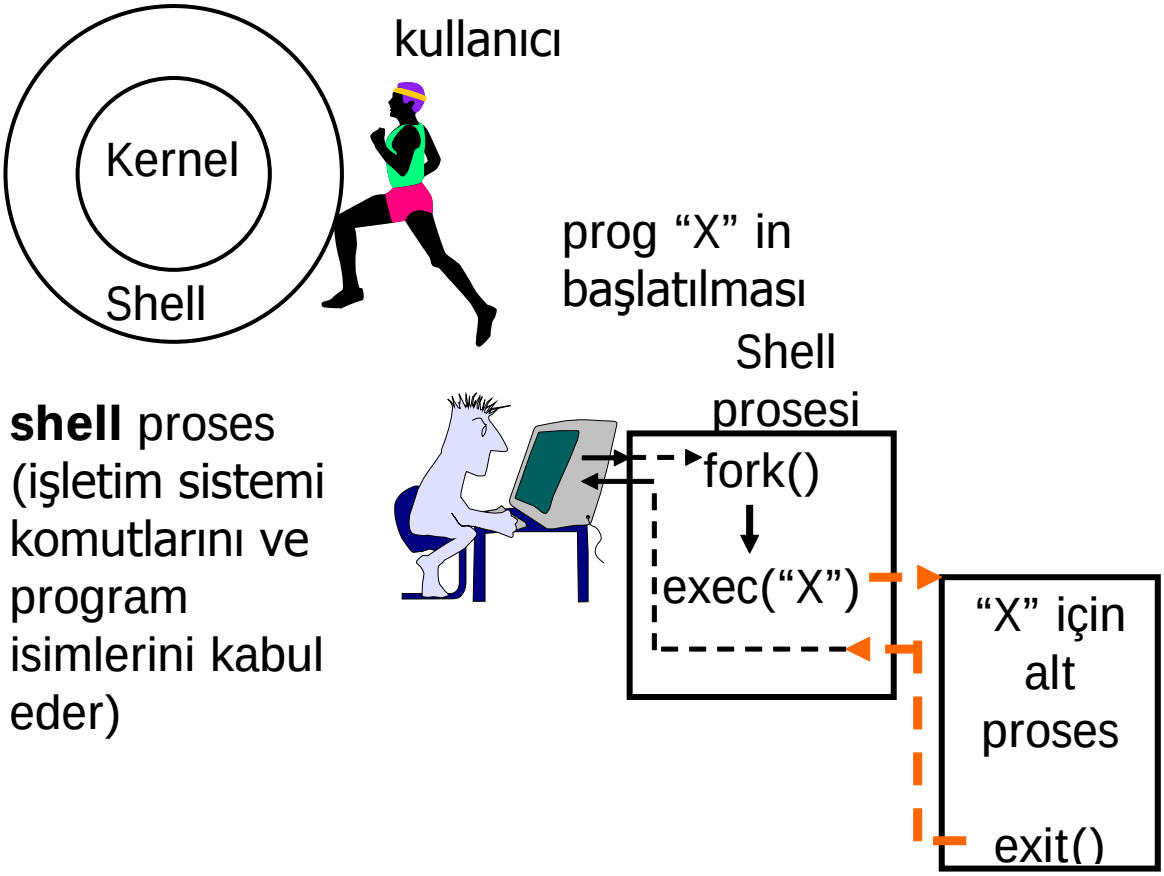
int system(const char *string) ;



Örnek:

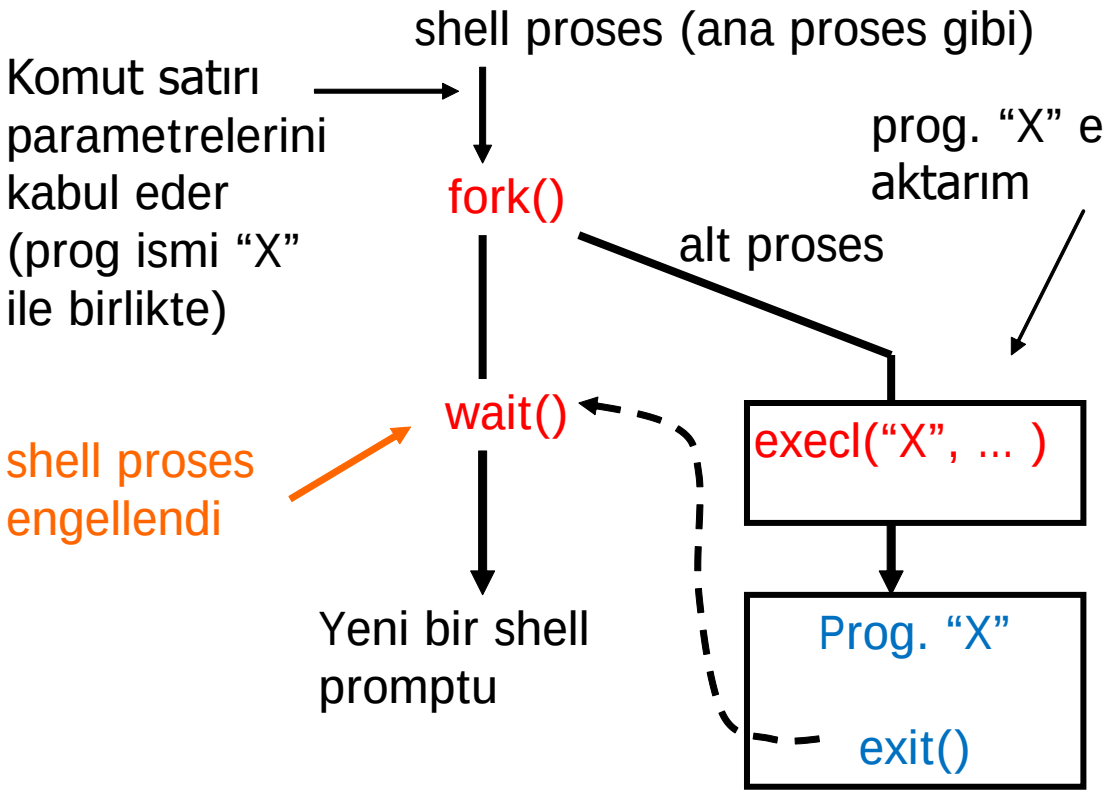
```
main()
{
   int status ;
   . . . .

   status = system( “yeni_prog par1 par2”) ;

   Dönen değerin analizi
 . . .
}
```

Kullanıcı programı shell prosesin bir alt prosesi olarak çalıştırılır

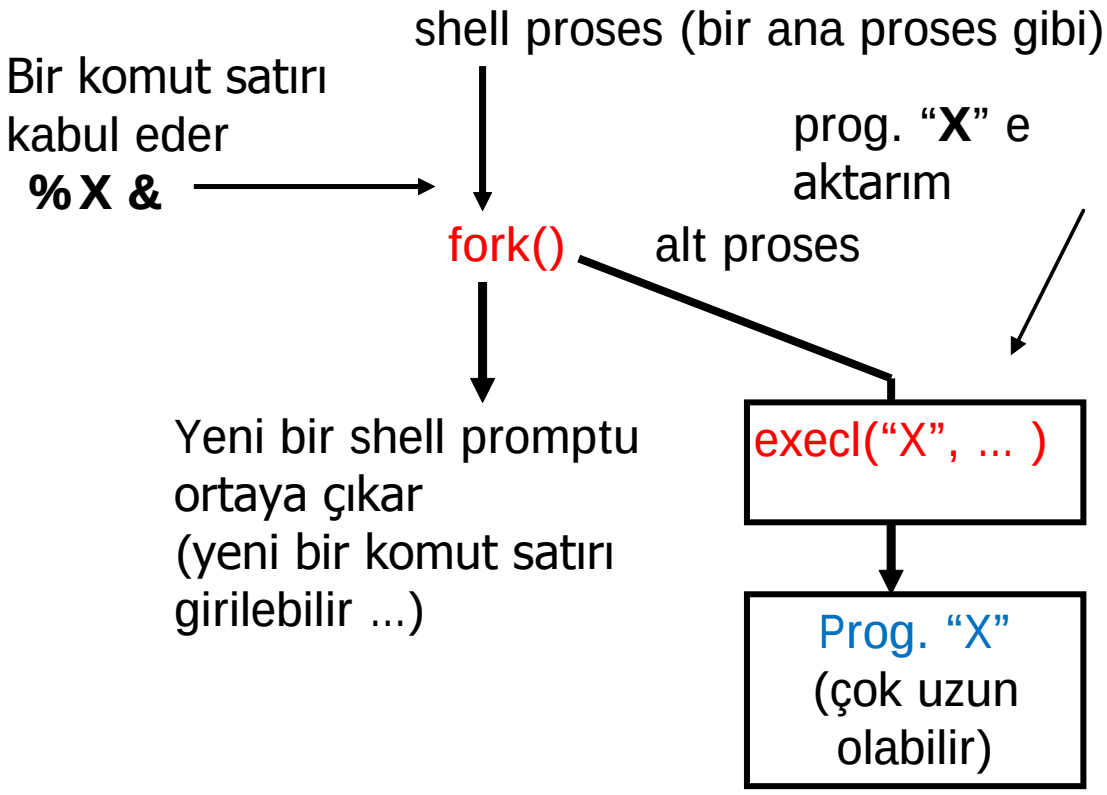


shell proses (işletim sistemi komutlarını ve program isimlerini kabul eder)

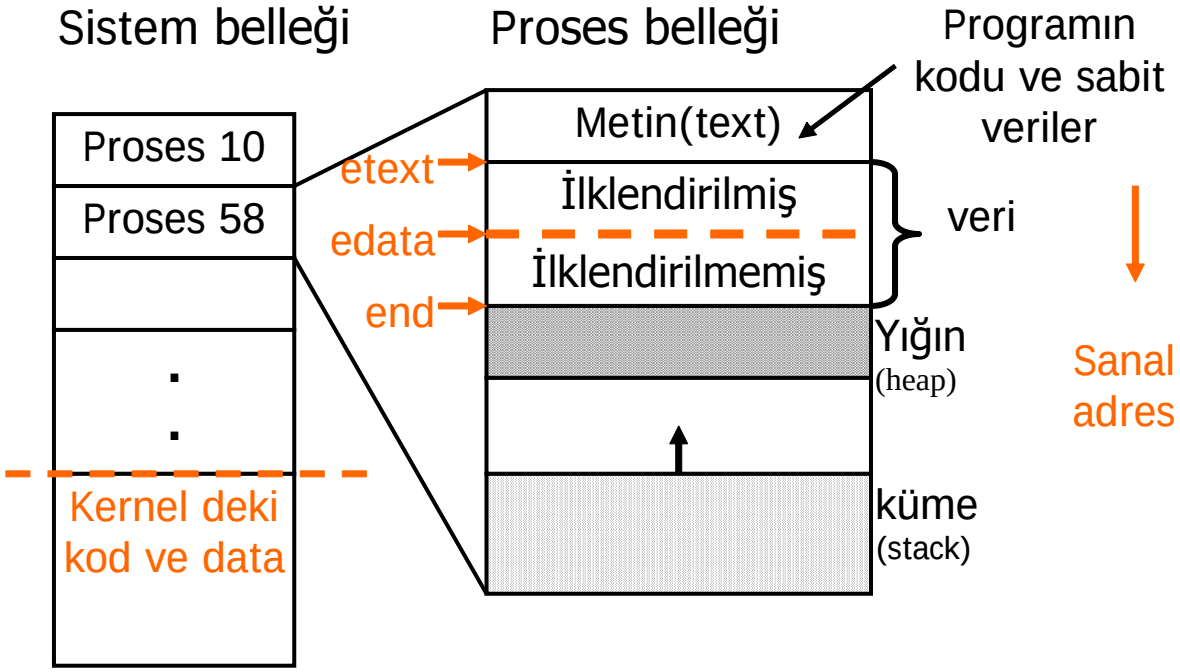
Shell proses
(kullanıcının programını ön planda çalıştırıyor)



Shell proses
(Kullanıcının programını arka planda çalıştırıyor)



System and process memory (virtual!)



Örnek:

```
#include . . .
char *ptr = “Hello\n”; /* static */  ← ilklendirilmiş alanda
char buf1[25];      ← ilklendirilmemiş alanda
main()
{ int show(char *) ;
  int i = 0 ;          /* otomatik */  ← küme alanında
  . . .
  for ( ; i < 3 ; ++i )  show( ptr ) ;
 }
int show( char *p)
{ char *buf2 ;  ← çağrılma esnasında küme alanında saklanır
  buf2 = malloc( (unsigned)(strlen(p)+1)) ; ← yığın alanından ayrılır
  strcpy(buf2, p) ;
  printf(“%s”, buf2) ;
  free( buf2 ) ;  ⟵   yığın alanından serbest bırakılır
  exit(0) ;
}
```

Programın sanal adresi