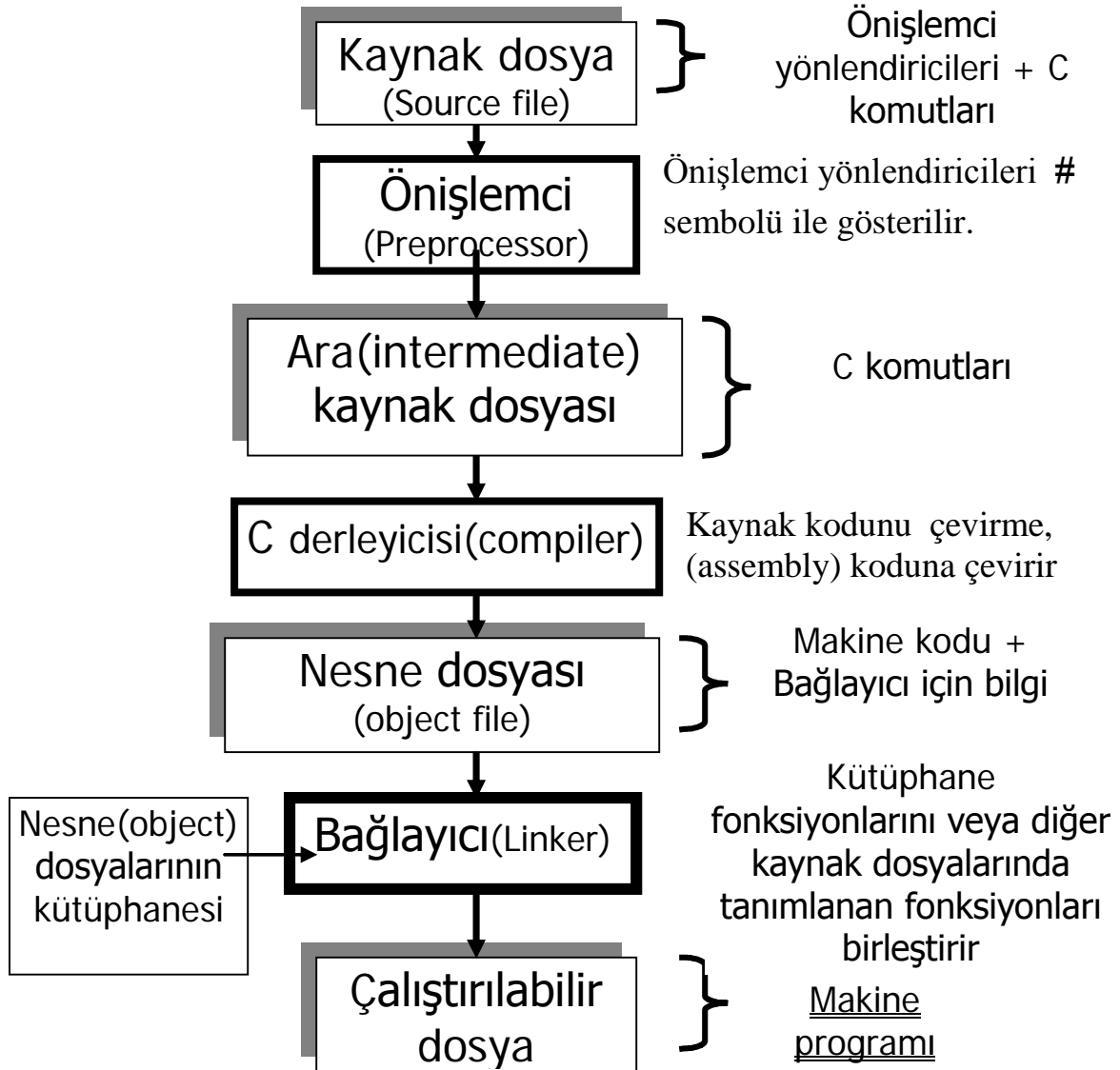


Bir C program uygulamasının adımları



Bazı mevcut C derleyicileri:

cc (or **gcc**) C compiler
CC (**g++**) C++ compiler

Bazı derleyici seçenekleri

- c** : Sadece derleme yapar.
Ex: %cc -c edit.c
- o** : Verilen isimde çalıştırılabilir dosya yaratır.
Ex: cc -o edit edit.c

UNIX de C program nasıl hazırlanır

Durum A: Programınız sadece bir kaynak dosyasından oluşuyorsa

1. Bir metin düzenleyiciyi(editor) çalıştırdıktan sonra (örneğin PICO), kaynak dosyasını yaratıp kaydedin(örneğin myprog.c).	%pico Pico ile çalışma
2. Programı derleyip bağlayarak çalıştırılabilir dosya yaratın (örneğin, myprog).	%cc -o myprog myprog.c
3. Programı çalıştırın	%myprog %myprog param1 param2 yada

Durum B: Programınız birkaç ayrı kaynak dosyasından oluşuyorsa (örneğin, module1.c, module2.c, module3.c)

1. Bir metin düzenleyicide dosyaları yaratın ve her birini kaydedin.	%cc -c module1.c
2. Her bir kaynak dosyası ayrı ayrı derlenip nesne dosyaları oluşturulur (module1.o, module2.o, module3.o)	%cc -c module2.c
3. Nesne dosyalarını bağlayarak çalıştırılabilir dosya yaratın.	%cc -c module3.c
	%cc -o myprog module1.o module2.o module3.o

Yada tek başına aşağıdaki komutu çalıştırın.

%cc -o **myprog** module1.c module2.c module3.c

Bir C programında main() fonsiyonun biçimleri

```
main( )  
{  
    Fonksiyonun gövdesi  
}
```

Komut satır
değişkeni(parametresi) olmadan

Çalıştırmak için:
%progismi

```
main(argc, argv)  
int argc ;  
char *argv[ ] ;  
{  
    Fonksiyonun gövdesi  
}
```

Bir veya daha fazla satır
değişkeni olunca

Çalıştırmak için:
%progismi param1 param2 ..

veya

main (int argc, char *argv[])

Örnek:

%time

← parametresiz

%mkdir **dizin ismi**

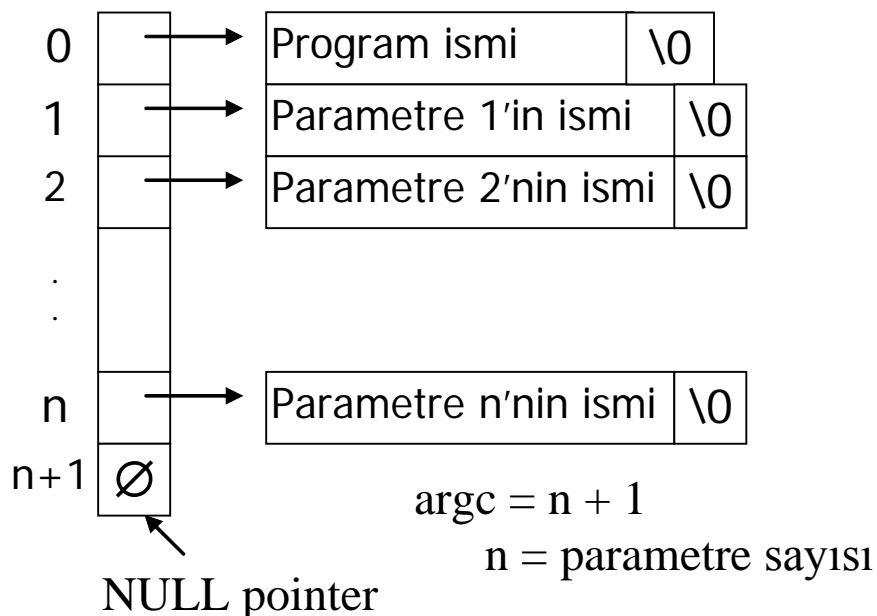
← parametrelili

- argc(değişken-argüman sayısı) komut satırındaki argümanların sayısı
 - argv(argüman vektörü) Tek boyutlu bir işaret dizilimini(array of pointers) işaret eder. Dizilimin her bir elemanı komut satırındaki bir karakter dizgi(character string) argümanını işaret eder.
- argv[0] Program ismini işaret eder
 - argv[i] i=1,2,..., argc-1 i. argümanı işaret eder ve
 - argv[argc] argümanların sonunu gösterir. Boş işaret(NULL pointer).

main() fonksiyonunun argümanlarını anlama

```
/* argc: komut satırındaki kelime-argüman sayısı */  
/* argv: argümanları işaret eden işaret dizilimi */  
int main (int argc, char *argv[] )  
{  
    .....  
}
```

*argv [] yapısı (Structure)



argv' ü programda kullanma:

argv[0] veya ***argv** – program ismini işaret eder

argv[1] – 1. parametreyi işaret eder

Not: Komut satırı argümanları her zaman karakter dizgisi (character string) olarak saklanırlar.

Örneğin: %myprog 35 56

iki parametre sırasıyla "35" ve "56" karakter dizgileridir. argv[1] ve argv[2] bu karakter dizgilerini işaret etmektedirler.

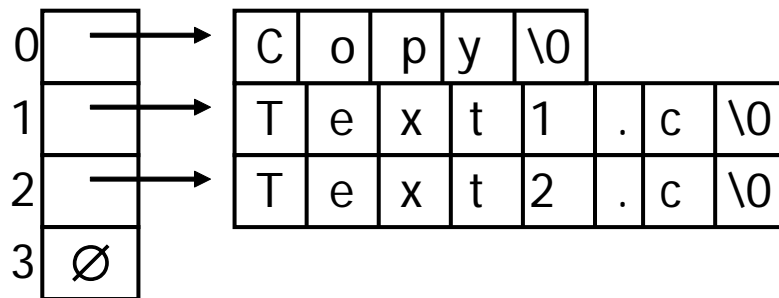
Örnek 1:

`% copy text1.c text2.c`

Burda

argc = 3

***argv [] yapısı** Yol isimlerini de içermektedir



argv[0] – "<yol(path)>/copy\0" işaret eder

argv[1] – "text1.c\0" işaret eder

argv[2] – "text2.c\0" işaret eder

argv[3] – Hiçi (NULL) işaret eder

Örnek 2: Aşağıda verilen komut satırı için

main (int argc, char *argv[])

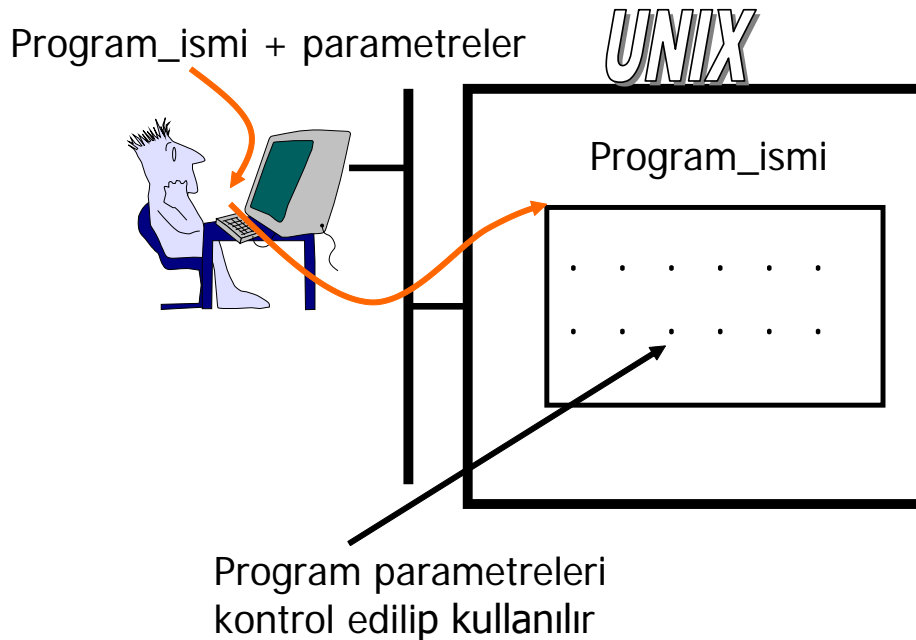
`a.out -s -o output input`

argv[] yapısı:

argv[0]	----->	a.out
argv[1]	----->	-s
argv[2]	----->	-o
argv[3]	----->	output
argv[4]	----->	input
argv[5]	----->	NULL

argc = 5

Komut satırı parametrelerinin C programında kullanımı



Örnek:

```
int main (int argc, char *argv[])
{
    int gün, yıl;
    /* Programın kullanımı */
    if (argc != 3)
    {
        printf("Kullanım: %s <gün> <yıl>\n", argv[0]);
        return 1;
    }
    gün = atoi(argv[1]); /* gerekli kütüphane dosyası (header file) <stdlib.h dir>*/
    yıl = atoi(argv[2]); /* verilen karakter dizilimini tam sayıya (int) çevirir */
    ....
}
```

Program parametresiz çalıştırılırsa

Kullanım: progismi <gün> <yıl> yazdırır

Komut satırı parametreleri programın içinde nasıl kullanılır

Örnek 1:

Aşağıda bir dosyayı başka dosyaya kopyalayan C programının bir parçası verilmiştir.

```
int main (int argc, char *argv[])
{
    ...
    f1 = open(argv[1], ...);
    f2 = creat(argv[2], ...);
    ...
}
```

Çalıştırılabilen(yürütülebilen) dosya oluşturulduktan sonra programı başlatmak için yazılan komut satırı:

%copy filename1 filename2

Programın
ismi

2 parametresi

Örnek 2:

```
/*kaynak dosya: myprog.c , çalıştırılabilir dosya: myprog */
#include <stdio.h>
int main ( int argc, char *argv[] )
{
    if ( argc < 2 )
    {
        printf( "Kullanım : %s parametre\n", argv[0] ) ;
        exit ( 1 ) ;
    }
    printf("Çalıştırılan program %s \n", argv[0] ) ;
    printf("Parametrelerin sayısı%d \n", argc-1 ) ;
    printf("Birinci parametre %s\n", argv[1] ) ;
    exit ( 0 ) ;
}
```

Komut satırı:

%myprog /* yanlış kullanım – parametre girmeden */

Çıktı(Output):

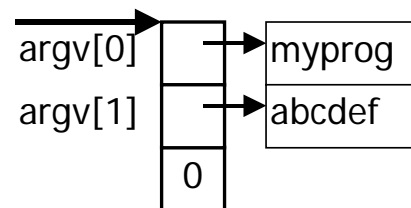
Kullanım: myprog parametre

Komut satırı:

%myprog abcdef /*doğru kullanım- parametre girerek */

Çıktı(Output):

Çalıştırılan program myprog
Parametrelerin sayısı 1
Birinci parametre abcdef



Programda olası değişiklikler:


argv[0] yerine → *argv
argv[1] yerine → *++argv

Örnek 3 :

```
/*Kaynak ismi: myprog.c ,  parametre: tam sayı */
#include <stdio.h>
int main ( int argc, char *argv[] )
{
    int p ;

    if ( argc < 2 )
    { printf( "Kullanım : %s parametre\n", argv[0] ) ;
      exit ( 1 ) ;
    }
    printf("Çalıştırılan program %s \n", argv[0] ) ;
    printf("Parametreleri %d \n", argc-1 ) ;

    p = atoi(argv[1]) ;
    printf("Birinci parametre %d\n", p ) ;
    exit ( 0 ) ;
}
```



Komut satırı:

%myprog 12

Çıktı:

.
.
Birinci parametre 12

More about command line parameters & options see the text book, Curry, pp47-49

Parametre sayılarını test eden bir örnek

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    for ( ; *argv ; ++argv )
        printf("%s\n", *argv ) ;
}
```

Komut satırı:

%benimprog bu bir testtir

Output:

benimprog
bu
bir
testtir

Örnek:

```
#include <stdio.h>

int main(int argc, char *argv[])
{ /* Komut satırındaki tüm argümanları yazdıran program */
    int i;

    printf("argc = %d\n\n",argc);
    for (i=0;i<argc;++i)
        printf("argv[%d]: %s\n",i, argv[i]);
}
```