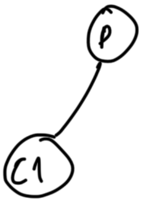
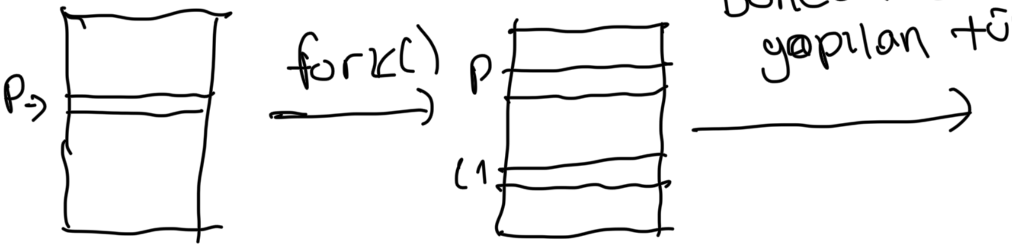
Pipes

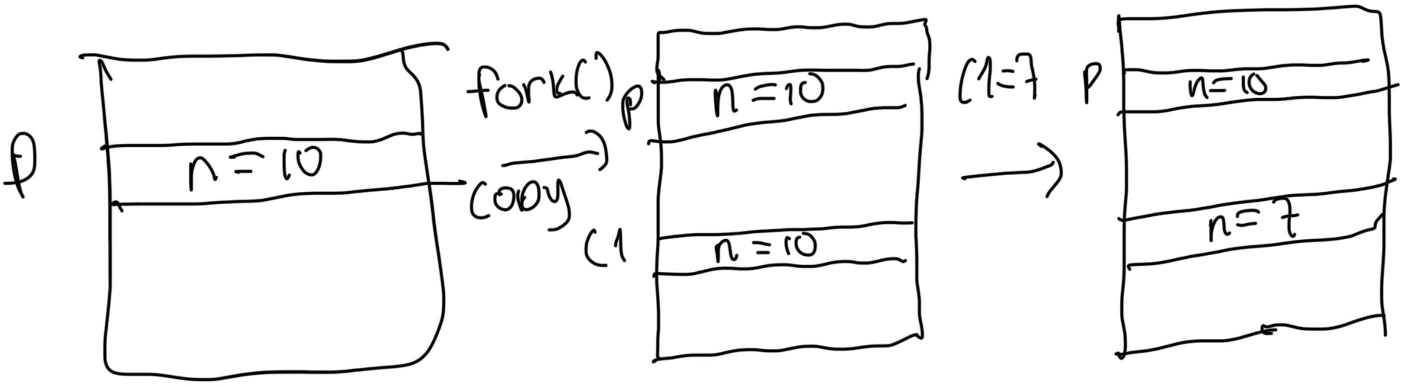


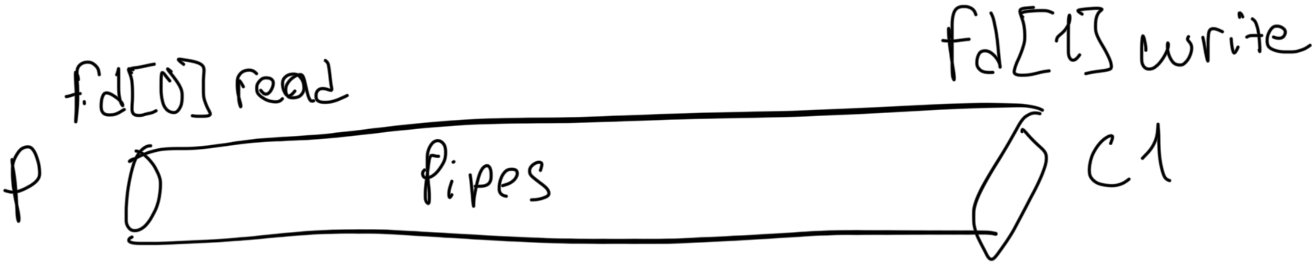
: Bütün memory Kopyalanıyor

fork()



Bundan sonra yapılan tüm işlemler bağımsız





Pipe'ı kesinlikle forktan önce tanımlamanız gerekmekte. Çünkü forktan önceki tüm memory Kopyalanmaktadır. Ve eğer siz forktan sonra pipe tanımlar iseniz o process'e özel pipe tanımlamış olursunuz

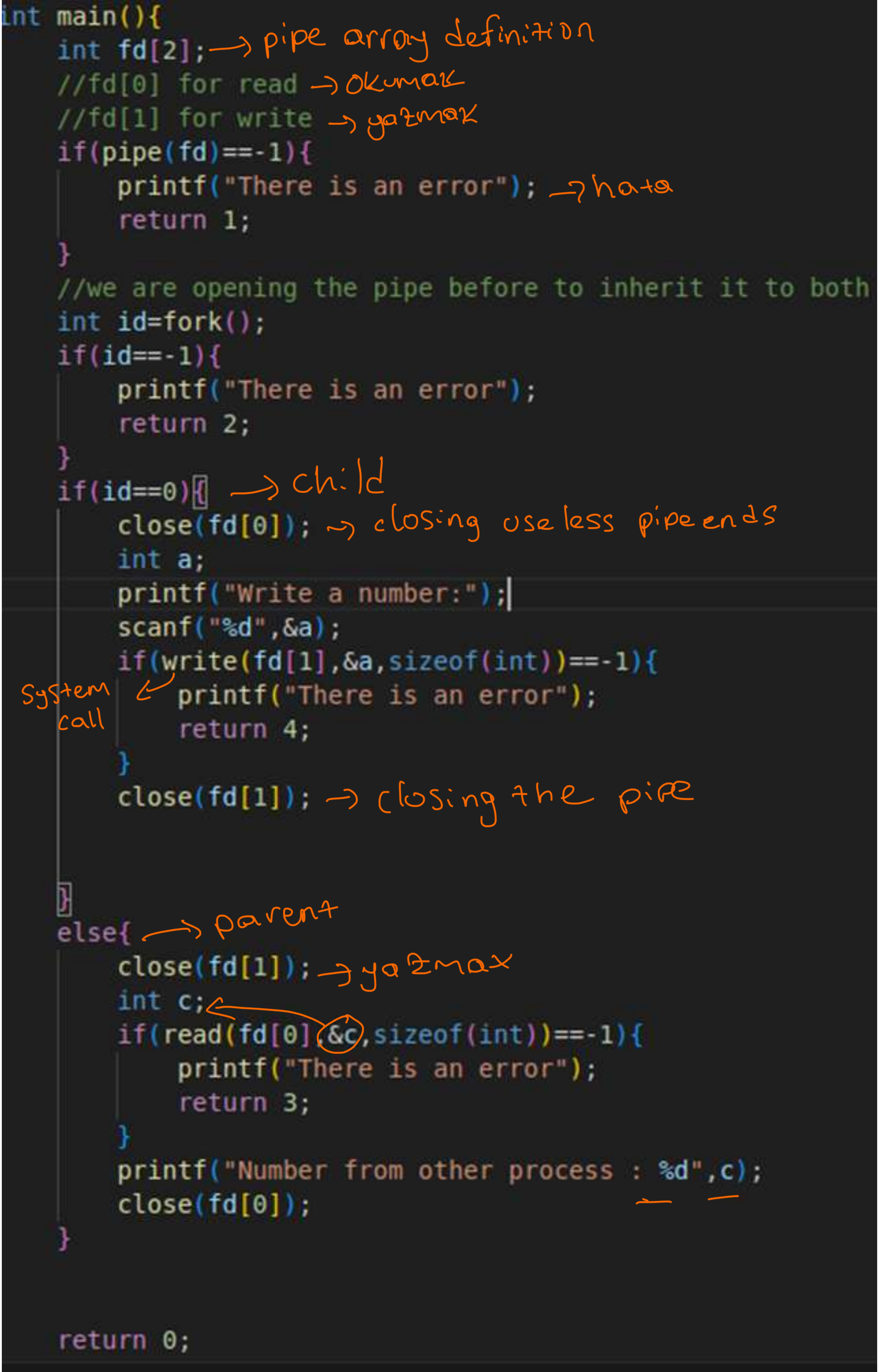
* write: write system call. Dosya yazmada da kullanılır pipe yazmada da.

write(Pipe ucu, değerin pointer, değerin büyüklüğü)

Returns number of bytes written

* read. read(pipe ucu, buffer pointer, değerin büyüklüğü)

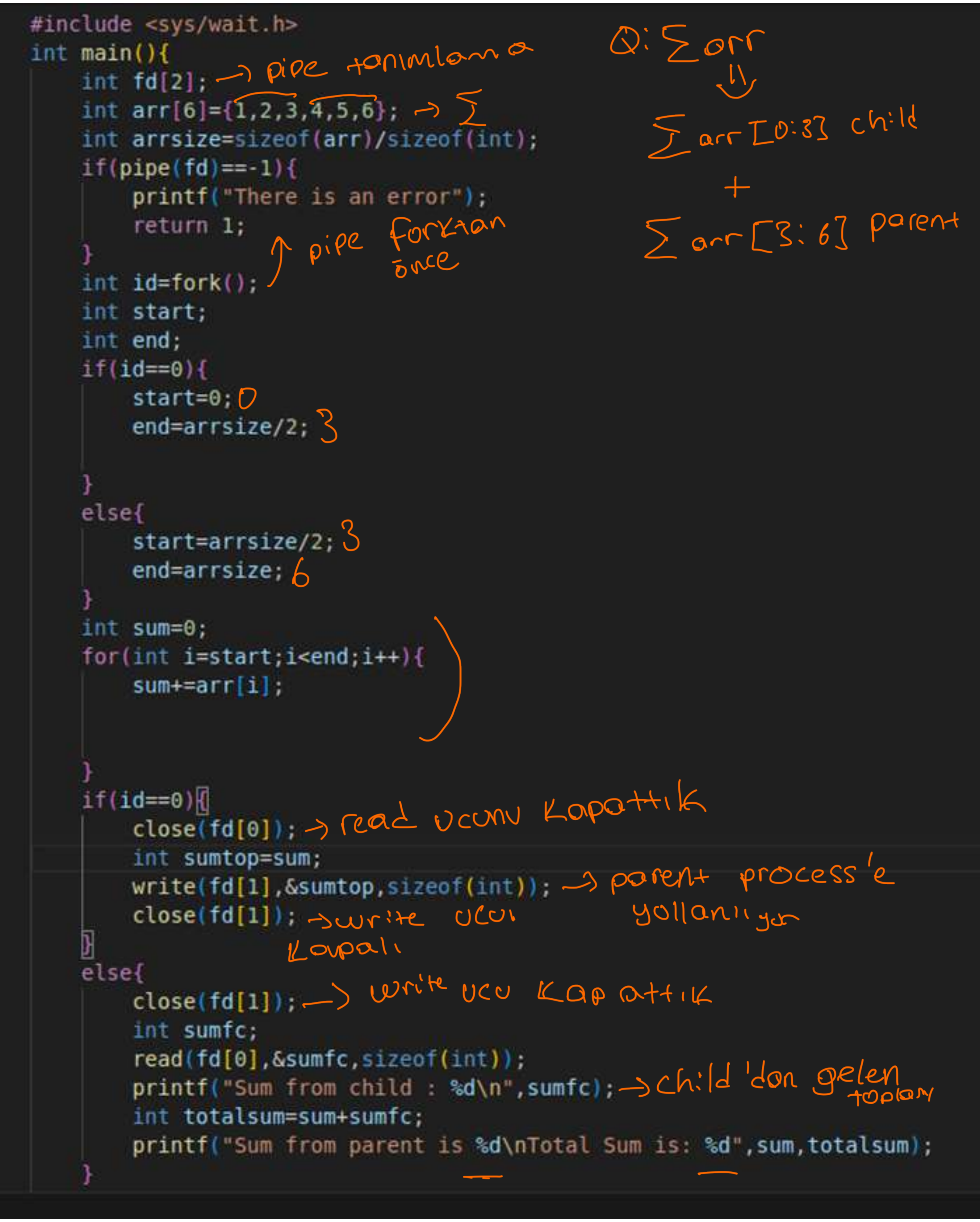
returns number of bytes read



```
int main(){
    int fd[2]; → pipe array definition
    //fd[0] for read → okumak
    //fd[1] for write → yazmak
    if(pipe(fd)==-1){
        printf("There is an error"); → hata
        return 1;
    }
    //we are opening the pipe before to inherit it to both
    int id=fork();
    if(id==-1){
        printf("There is an error");
        return 2;
    }
    if(id==0){ → child
        close(fd[0]); → closing useless pipe ends
        int a;
        printf("Write a number:");
        scanf("%d",&a);
        if(write(fd[1],&a,sizeof(int))==-1){
system call ←
            printf("There is an error");
            return 4;
        }
        close(fd[1]); → closing the pipe


    }
    else{ → parent
        close(fd[1]); → yazmax
        int c;
        if(read(fd[0],&c,sizeof(int))==-1){
            printf("There is an error");
            return 3;
        }
        printf("Number from other process : %d",c);
        close(fd[0]);
    }


    return 0;
```



```c
#include <sys/wait.h>
int main(){
    int fd[2];
    int arr[6]={1,2,3,4,5,6};
    int arrsize=sizeof(arr)/sizeof(int);
    if(pipe(fd)==-1){
        printf("There is an error");
        return 1;
    }
    int id=fork();
    int start;
    int end;
    if(id==0){
        start=0;
        end=arrsize/2;

    }
    else{
        start=arrsize/2;
        end=arrsize;
    }
    int sum=0;
    for(int i=start;i<end;i++){
        sum+=arr[i];


    }
    if(id==0){
        close(fd[0]);
        int sumtop=sum;
        write(fd[1],&sumtop,sizeof(int));
        close(fd[1]);
    }
    else{
        close(fd[1]);
        int sumfc;
        read(fd[0],&sumfc,sizeof(int));
        printf("Sum from child : %d\n",sumfc);
        int totalsum=sum+sumfc;
        printf("Sum from parent is %d\nTotal Sum is: %d",sum,totalsum);
    }
```

Q: $\sum$ arr

⇓

$\sum$ arr [0:3] child

+

$\sum$ arr [3:6] parent

→ pipe tanımlama

→ $\sum$

↑ pipe forktan önce

0

3

3

6

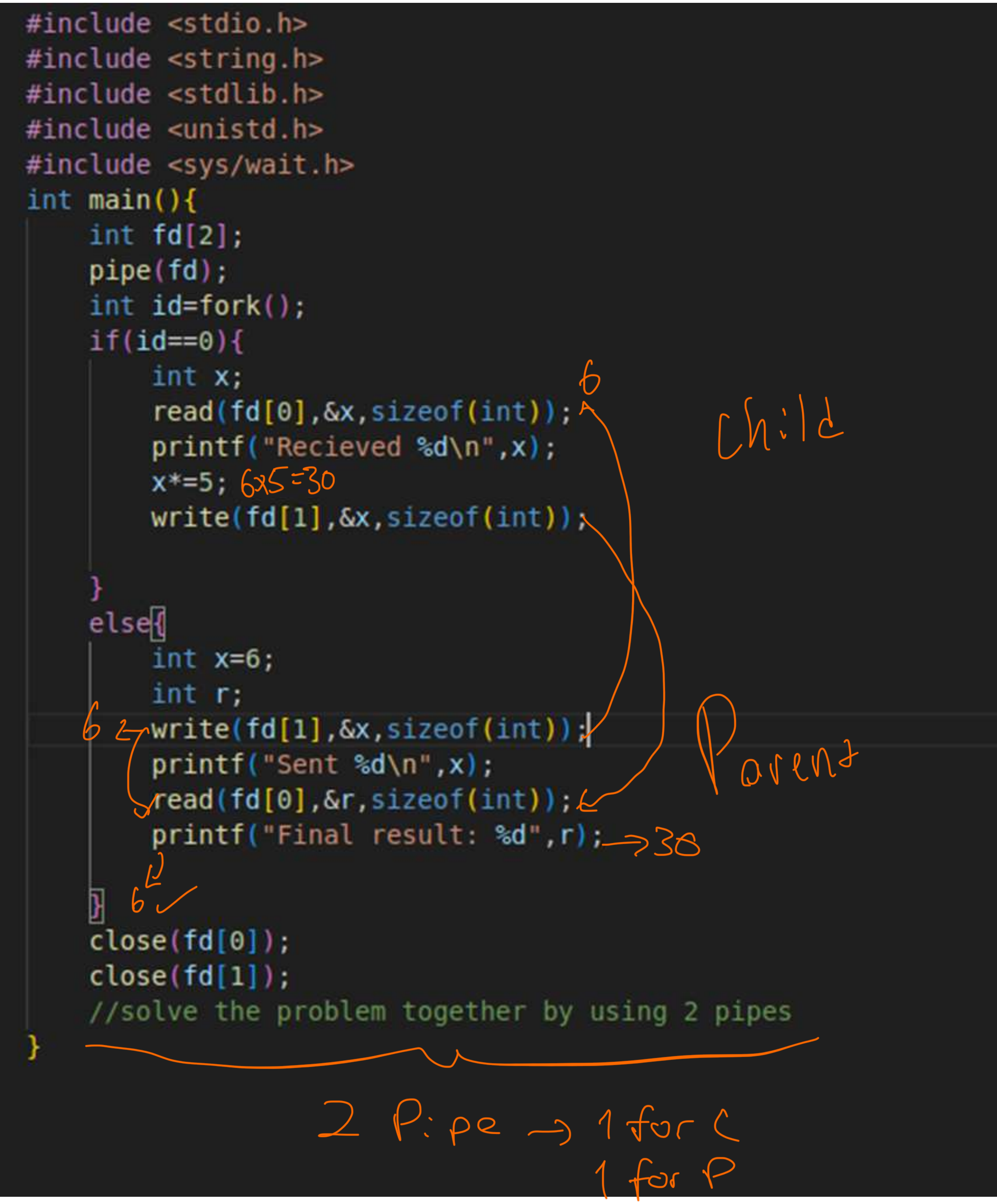
→ read ucunu kapattık

→ parent process'e yollanıyor

→ write ucu kapalı

→ write ucu kapattık

→ child'dan gelen toplam

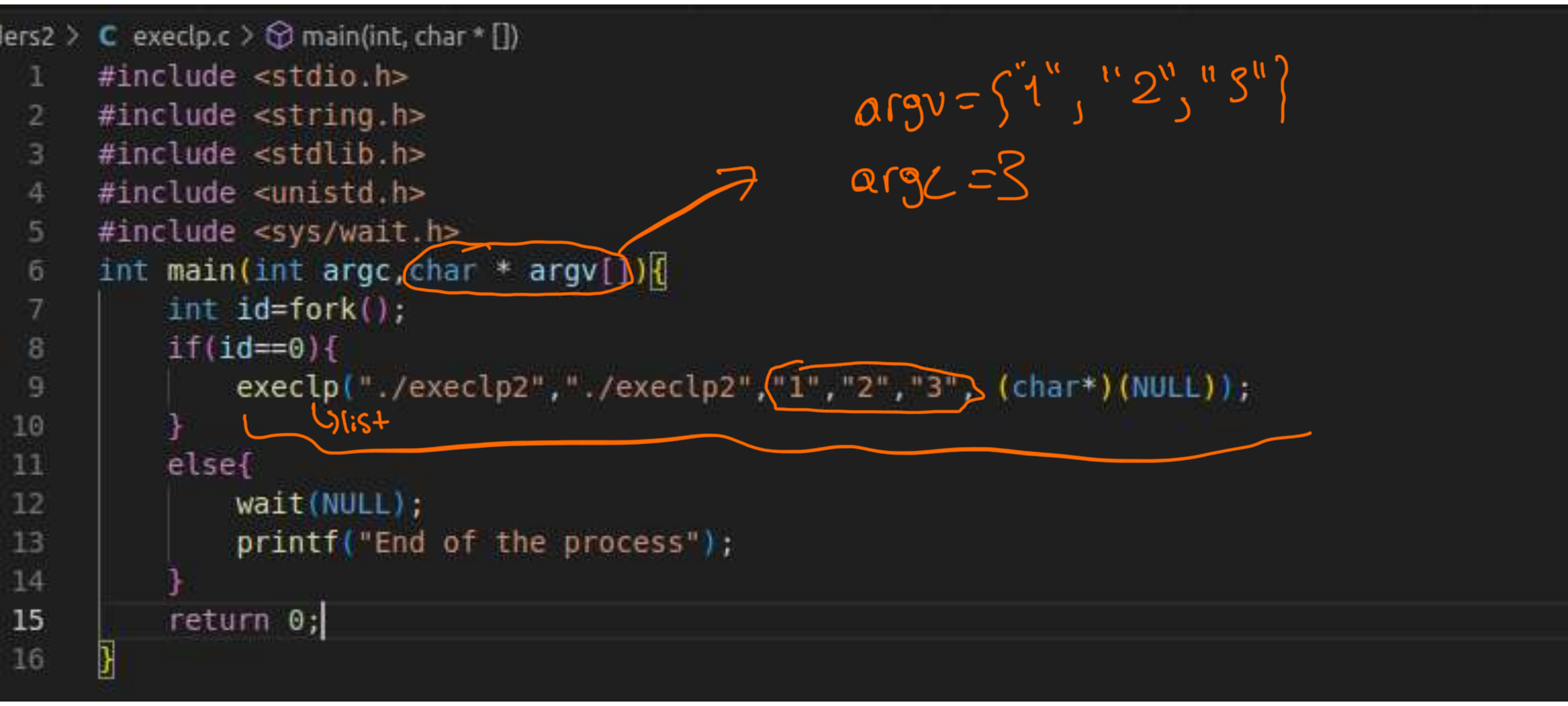


```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main(){
    int fd[2];
    pipe(fd);
    int id=fork();
    if(id==0){
        int x;                                         6
        read(fd[0],&x,sizeof(int));                         Child
        printf("Recieved %d\n",x);
        x*=5; 6x5=30
        write(fd[1],&x,sizeof(int));

    }
    else{
        int x=6;
        int r;
  6 2   write(fd[1],&x,sizeof(int));                      Parent
        printf("Sent %d\n",x);
        read(fd[0],&r,sizeof(int));
        printf("Final result: %d",r); → 30

    } 6 ✓
    close(fd[0]);
    close(fd[1]);
    //solve the problem together by using 2 pipes
}
```

2 Pipe → 1 for C
         1 for P

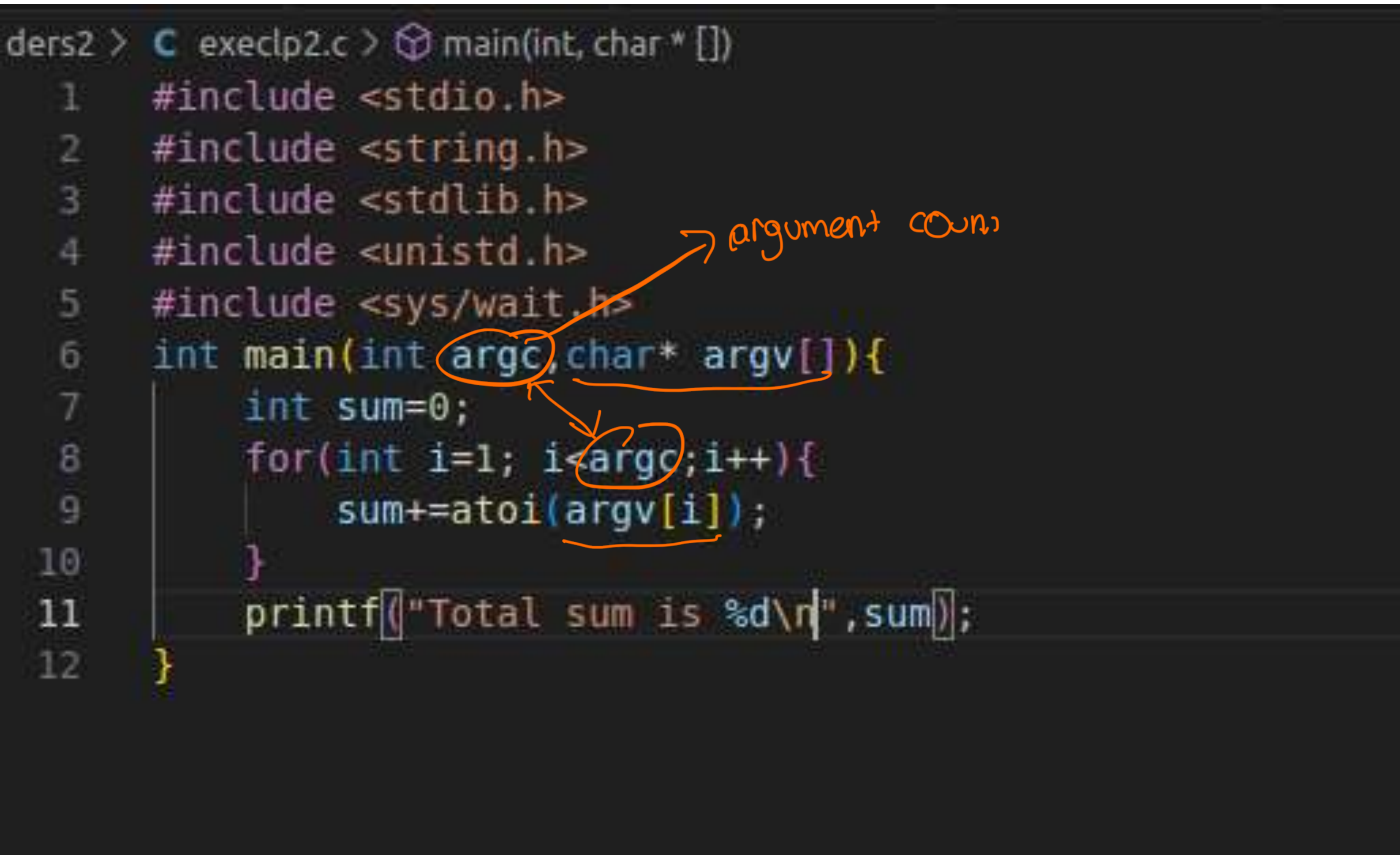


```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main(int argc,char * argv[]){
    int id=fork();
    if(id==0){
        execlp("./execlp2","./execlp2","1","2","3", (char*)(NULL));
    }
    else{
        wait(NULL);
        printf("End of the process");
    }
    return 0;
}
```

argv={"1","2","3"}

argc=3

list



```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main(int argc,char* argv[]){
    int sum=0;
    for(int i=1; i<argc;i++){
        sum+=atoi(argv[i]);
    }
    printf("Total sum is %d\n",sum);
}
```

argument count