Comanda env ne arata variabilele de mediu. In aceste variabile se gaseste PATH-ul. Cand executam o com

Comanda exec :

Caută în PATH

ARGUMENTE	DA	NU
Vector	char *a[] = {"grep", "-E", "abc", "a.txt", NULL}; execvp("grep", a);	char *a[] = {"/bin/grep", "-E", "abc", "a.txt", NULL}; execv("/bin/grep", a);
Lista	execlp("grep", "grep", "-E", "abc", "a.txt", NULL);	execl("/bin/grep", "/bin/grep", "-E", "abc", "a.txt", NULL);

Exemplu :

```c
#include <stdio.h>
#include <unistd.h>

int main(){
        printf("a\n");
        execlp("echo","echo","b",NULL);
        printf("c\n");
        return 0;
}
```

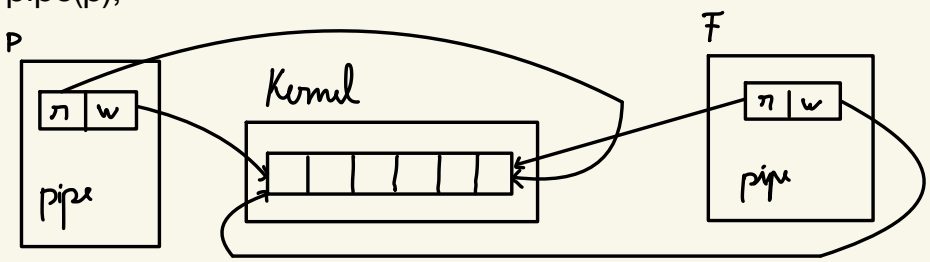
Daca instructiunea exec s-a executat cu succes atunci ceea ce se afla dupa comanda nu se va mai executa. Daca dorim ca programul sa nu se mai opreasca dupa executia instructiunii exec putem scrie programul astfel :

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){
        printf("a\n");
        if (fork() == 0){
                if(execlp("echo","echo","b",NULL) < 0){
                        exit(1);
                }
        }
        printf("c\n");
        wait(0);
        return 0;
}
```

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){
        int a[4] = {1,2,3,4}

        if (fork() == 0){
                a[2] += a[3];
                exit(0);
        }
        a[0] += a[1];
        wait(0);
        a[0] += a[2];

        printf("%d\n", a[0]);
        return 0;
}
```

Aici procesul fiu creat in if prin fork() va avea un tablou separat fata de procesul parinte, deci ceea ce se intampla in if in procesul fiu nu va fi salvat in procesul parinte rezultand in output 6.

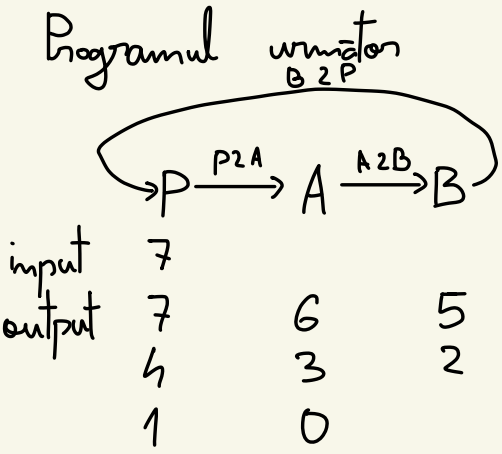
PIPE
int p[2]
pipe(p);



Reguli :

- prin PIPE pot sa comunice doar procesele care mostenesc descriptorii de acces in pipe
- inchideti capetele PIPE-ului cat mai curand posibil, daca nu, programul va ingheta

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){
        int a[4] = {1,2,3,4}
        int p[2];
        pipe(p)
        if (fork() == 0){
                close(p[0]);
                a[2] += a[3];
                write(p[1],&a[2],sizeof(int));
                close(p[1]);
                exit(0);
        }
        close(p[1]);
        a[0] += a[1];
        read(p[0],&a[2],sizeof(int));
        close(p[0]);
        wait(0);
        a[0] += a[2];

        printf("%d\n", a[0]);
        return 0;
}
```



input	7		
output	7	6	5
	4	3	2
	1	0	