28.03.2019

Curs. SO ♥

```
void wait_child (int sig) { wait(0);}
signal (SIGCHLD, wait_child); // signal(SIGCHLD, SIG_IGN)
for(;;) {

    get_request
    if (fork()==0) {

        process_req
        send_reply
        exit(0);
    }
}
```

// nohup → ignoră semnale de hangup

funcția signal nu face nimic !!

signal (semnal, func)
funcția care trimite un semnal este kill (pid, semnal)

nu.c

```
#include <stdio.h>
#include <signal.h>
int main( int argc, char **argv ){

void nu ( int sig ) {
    printf ("Nu mean!\n");
}

int main( int argc, char **argv ){

    signal (SIGINT), nu);
    while (1);
    return 0;
}
```

Kill -9 pid

man signal → see also
nu mean ? signal : toate semnalele
&bg background SIGSTOP
CTRL-Z fg → mi-am (SIGCONT)

SIGKILL nu se poate suprascrie

→ ..

execl
execlp
execle
execv
execvp
execvpe

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

int main(int argc, char** argv){

    execlp("grep", "grep", "/an1/gr011/", "/etc/passwd", NULL);
    printf("If grep is in the PATH, then execlp succeeds, and this will never be printed.\n");
    return 0;
}
```

PATH → ★ fiecare program pe care îl ● pornim are o locație pe disc

/bin/grep

PATH = var. de mediu → locație ~~unde~~ cu executabile

toate variantele de exec cu (p) caută în PATH

	SEARCH PATH: Yes	SEARCH PATH: No
Array	char *a[] = {"grep", ...}; execvp("grep", a);	char *a[] = {"/bin/grep", ...}; execv("/bin/grep", a);
List	execlp("grep", "grep", "/an1/gr011/", "/etc/passwd", NULL);	execl("/bin/grep", "/bin/grep", "/an1/gr011/", "/etc/passwd", NULL);

"V = a"
"a = 5" } → VARIABILE DE MEDIU

exec_grep.c

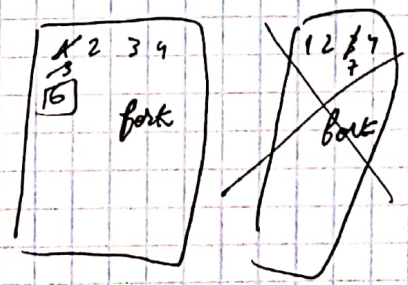
```
#include <stdio.h>
#include <unistd.h>
int main(int argc, char ** argv){
    execlp("grep","grep","-i","[aeiou][aeiou][aeiou]","/etc/passwd",NULL);
    printf("=======");  //asta nu se mai rulează dacă execlp merge
    return 0;
}
```

→ folosim exec în procesul fiu ca să nu ne pierdem codul

```
#include<sys/wait>
#include<unistd.h>
#include<stdio.h>
#include<unistd.h>
#include<sys/type>
int main(int argc, char** argv){
    if(fork()==0){
        execlp("grep","grep","-i","[aeiou][aeiou][aeiou]","/etc/passwd",NULL);
        exit(0); //dacă execlp nu se execută, pentru siguranță
    }
    wait(0);
    printf("=========");
    printf("=========");
    return 0;
}
```

num.c

```
#include<...
int main(int argc, char** argv){
    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));
        exit(0);
    }
    close(p[1]);
    read(p[0],&a[2],sizeof(int));
    wait(0); close(p[0]);
    a[0]+=a[1];
    printf("%d",a[0]); //afișează 6, pentru că procesul fiu are memorie proprie care
    return 0;          // nu revine în procesul părinte.
}
```



Pipe = buffere în memorie

read → ne promite că citește cât scrie