

Operációs rendszerek BSc

2. Gyak.

2022. 03. 18.

Készítette:

Kopecskó Zsolt Bsc

Mérnökinformatikus

IVO46O

Miskolc, 2022

1. feladat – A system() rendszerhívással hajtson végre létező és nemlétező parancsot, és vizsgálja a visszatérési értéket.

(IVO46O_1fel.c)

```
#include <stdlib.h>
```

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

```
#include <signal.h>
```

```
int main(){
```

```
    int sc;
```

```
    char parancs[10];
```

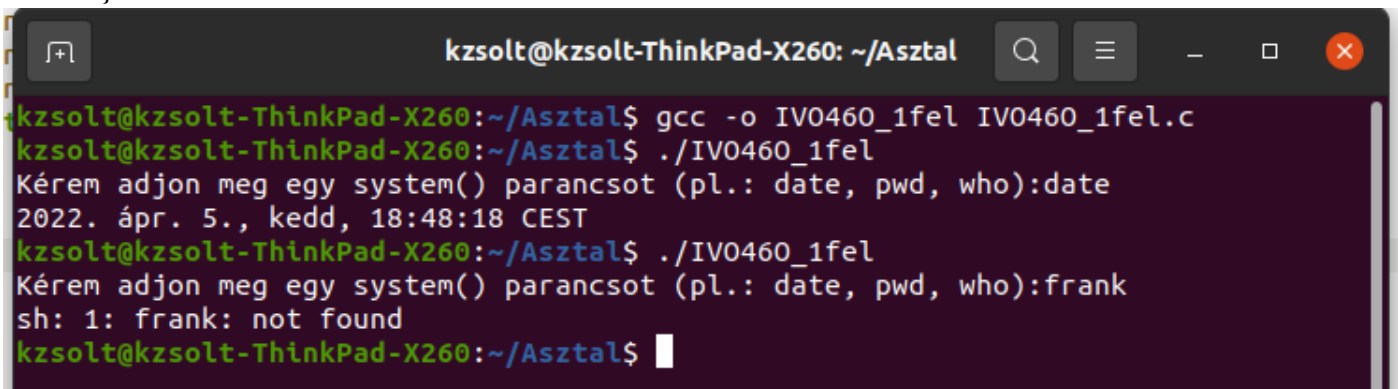
```
    printf("Kérem adjon meg egy system() parancsot (pl.: date, pwd, who):");
```

```
    scanf("%s", parancs);
```

```
    sc=system(parancs);
```

```
    return 0;
```

```
}
```



```
kzsolt@kzsolt-ThinkPad-X260: ~/Asztal
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$ gcc -o IVO46O_1fel IVO46O_1fel.c
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$ ./IVO46O_1fel
Kérem adjon meg egy system() parancsot (pl.: date, pwd, who):date
2022. ápr. 5., kedd, 18:48:18 CEST
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$ ./IVO46O_1fel
Kérem adjon meg egy system() parancsot (pl.: date, pwd, who):frank
sh: 1: frank: not found
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$
```

A date system call hatására a program kiírta az rendszeridőt!

Amikor hibás system call-t adtam meg („frank”), akkor azt írta ki, hogy ez a parancs nem található.

2. feladat – Írjon programot, amely billentyűzetről bekér UNIX parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre.

(IVO46O_2fel.c)

```
#include <stdlib.h>
```

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

```
#include <signal.h>
```

```
int main(){
```

```
    int sc;
```

```
    char parancs[10];
```

```
    do{
```

```
        printf("\nKérem adjon meg egy system() parancsot (pl.: date, pwd, who),\n vagy a kilépéshez  
ctr+c :");
```

```
        scanf("%s", parancs);
```

```
        sc=system(parancs);
```

```
    }while(parancs[10]!=SIGQUIT);
```

```
    return 0;
```

```
}
```

```
kzsolt@kzsolt-ThinkPad-X260: ~/Asztal
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$ gcc -o IV0460_2fel IV0460_2fel.c
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$ ./IV0460_2fel

Kérem adjon meg egy system() parancsot (pl.: date, pwd, who),
vagy a kilépéshez ctr+c :date
2022. ápr. 5., kedd, 18:44:11 CEST

Kérem adjon meg egy system() parancsot (pl.: date, pwd, who),
vagy a kilépéshez ctr+c :pwd
/home/kzsolt/Asztal

Kérem adjon meg egy system() parancsot (pl.: date, pwd, who),
vagy a kilépéshez ctr+c :who
kzsolt  :0          2022-04-05 17:20 (:0)

Kérem adjon meg egy system() parancsot (pl.: date, pwd, who),
vagy a kilépéshez ctr+c :^C
kzsolt@kzsolt-ThinkPad-X260:~/Asztal$
```

A date rendszerhívás kiírja az aktuális rendszeridőt.

A pwd rendszerhívás kiírja az aktuális munkakönyvtár elérési útvonalát.

A who rendszerhívás kiírja a bejelentkezett felhasználót.

Égészen addig végzi az adatbekérést a felhasználótól, míg nem érzékeli a SIGQUIT jelet.

3. Készítsen egy parent.c és egy child.c programot. A parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre 10-szer.

(parent.c)

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

```
void parentTask() {
    printf("Sikeresen lefutott a gyermek process.\n");
}

int main(void) {
    pid_t pid = fork();

    if(pid == 0) {
        childTask();
        exit(0);
    }
    else if(pid > 0) {
        wait(NULL);
        parentTask();
    }
    else {
        printf("Nem sikerült létrehozni a gyermek processzt.");
    }
}
```

```

return 0;
}

(child.c)
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

void childTask() {

    for(int i=0; i<10; i++){
        printf("[%d]: Kopecskó Zsolt IVO46O\n",i+1);
    }
    exit(0);
}

```

```

kzsolt@kzsolt-ThinkPad-X260: ~/os/gyak2
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ gcc child.c -c
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ gcc child.o parent.c -o parent_fut
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ ./parent_fut
[1]: Kopecskó Zsolt IVO46O
[2]: Kopecskó Zsolt IVO46O
[3]: Kopecskó Zsolt IVO46O
[4]: Kopecskó Zsolt IVO46O
[5]: Kopecskó Zsolt IVO46O
[6]: Kopecskó Zsolt IVO46O
[7]: Kopecskó Zsolt IVO46O
[8]: Kopecskó Zsolt IVO46O
[9]: Kopecskó Zsolt IVO46O
[10]: Kopecskó Zsolt IVO46O
Sikeresen lefutott a gyermek process.
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$

```

A parent.c-ben a fork() rendszerhívással létrehozunk egy gyermek processzt. Sikeres gyermek processz létrehozása esetén 0 értéket ad vissza, ha nem sikerül a létrehozás, akkor -1 értékkel tér vissza. Ha a sikeres volt a fork, akkor meghívásra és végrehajtásra kerül a child.c-ben lévő függvény. Miután sikeresen lefut a gyermek process, akkor az 0 értékkel kilép a metódusból. A szülő processz erre a 0 értékre vár. Amint megkapja, akkor lefut a szülő process is.

4. A fork() rendszerhívással hozzon létre egy gyermek processt és abban hívjon meg egy exec családbeli rendszerhívást (pl. execlp). A szülő várja meg a gyermek lefutását!

```

(IVO46O_4fel.c)
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

```

```

int main(void) {
    pid_t pid = fork();

```

```

if(pid == 0) {
    char *prog = "ls";
    char *arg1 = "-lh";
    char *arg2 = "/home/kzsolt/os/gyak2";
    execlp(prog, prog, arg1, arg2, NULL);
    exit(0);
}
else if(pid > 0) {
    wait(NULL);
    printf("Sikeresen lefutott a gyermek process.\n");
}
else {
    printf("Nem sikerült létrehozni a gyermek processzt.");
}

return 0;
}

```

```

kzsolt@kzsolt-ThinkPad-X260: ~/os/gyak2
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ gcc IVO460_4fel.c -o IVO460_4fel
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ ./IVO460_4fel
összesen 208K
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 4 18:18 a.out
-rw-rw-r-- 1 kzsolt kzsolt 230 máj 4 18:11 child.c
-rw-rw-r-- 1 kzsolt kzsolt 1,8K máj 4 18:08 child.o
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 5 19:48 combined
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 5 18:48 IVO460_1fel
-rw-rw-r-- 1 kzsolt kzsolt 265 ápr 5 18:47 IVO460_1fel.c
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 6 18:32 IVO460_24fel
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 6 18:32 IVO460_2fel
-rw-rw-r-- 1 kzsolt kzsolt 331 ápr 6 18:32 IVO460_2fel.c
-rwxrwxr-x 1 kzsolt kzsolt 17K máj 4 18:21 IVO460_4fel
-rw-rw-r-- 1 kzsolt kzsolt 516 máj 4 18:20 IVO460_4fel.c
-rwxrwxr-x 1 kzsolt kzsolt 17K ápr 6 18:53 IVO460_5fel
-rw-rw-r-- 1 kzsolt kzsolt 581 ápr 6 18:54 IVO460_5fel.c
-rwxrwxr-x 1 kzsolt kzsolt 17K máj 4 17:59 parent
-rw-rw-r-- 1 kzsolt kzsolt 472 ápr 5 19:59 parent.c
-rwxrwxr-x 1 kzsolt kzsolt 17K máj 4 18:09 parent_fut
Sikeresen lefutott a gyermek process.
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$

```

A parent.c-ben a fork() rendszerhívással létrehozunk egy gyermek processzt. Sikeres gyermek processz létrehozása esetén 0 értéket ad vissza, ha nem sikerül a létrehozás, akkor -1 értékkel tér vissza. Ha a sikeres volt a fork, akkor végrehajtásra kerül a gyermek processben deklarált execlp rendszerhívás. A függvénynek a paraméterein keresztül megadtuk, hogy a listázza ki a megadott könyvtárban lévő file-okat. Miután sikeresen lefut a gyermek process, akkor az 0 értékkel kilép a metódusból. A szülő processz erre a 0 értékre vár. Amint megkapja, akkor lefut a szülő process is.

5. A fork()rendszerhívással hozzon létre gyermekeket, várja meg és vizsgálja a befejezési állapotokat. (IVO46O_5fel.c)
- ```

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

```

```

int main(void) {
 pid_t pid = fork();
 pid_t pid1, pid2;
 int a;
 if(pid == 0) {
 printf("Szülő pid: [%d] ---> Gyerek pid: [%d]\n",getppid(), getpid());
 exit(0); //kilép a metódusból és 0 értéket ad vissza
 printf("vajon ez megjelenik? Remélem nem!");
 }
 else if((pid1=fork()) == 0){
 printf("Szülő pid: [%d] ---> Gyerek pid: [%d]\n",getppid(), getpid());
 abort(); //kilép a metódusból de nem ad vissza értéket.
 printf("vajon ez megjelenik? Remélem nem!");
 }else if((pid2=fork()) == 0){ // a nullával való osztás végett a
 int x=6; // a metódus le se fut.
 int y=0;
 printf("lássuk lehet-e nullával osztani");
 int z=x/y;
 printf("z=[%d]\n",z);
 }
 wait(NULL);
 printf("Sikeresen lefutott a gyermek process.\n");
 return 0;
}

```

```

kzsolt@kzsolt-ThinkPad-X260: ~/os/gyak2
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$ gcc IV0460_5fel.c -o IV0460_5fel
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$./IV0460_5fel
Szülő pid: [9492] ---> Gyerek pid: [9494]
Szülő pid: [9492] ---> Gyerek pid: [9493]
Sikeresen lefutott a gyermek process.
kzsolt@kzsolt-ThinkPad-X260:~/os/gyak2$

```

Az `exit(0)` hatására a gyerek befejezi a futást és 0 értéket ad vissza a szülőnek. `Break()` hatására gyermek befejezi a futását de értéket nem vissza a szülőnek. A nullával való osztás végett a gyermek el sem kezd futni.

6. Adott a következő ütemezési feladat, amit a FCFS, SJF és RR ütemezési algoritmus használatával készítsen el.  
(ivo46o\_6fel.pdf)