

Operációs rendszerek BSc

10.Gyak

2022.05.12.

Készítette:

Tucsa Eszter Boglárka
Mérnökinformatikus BSc
G2QWPO

1.feladat:

„1. Az előadáson bemutatott mintaprogram alapján készítse el a következő feladatot.

Adott egy rendszerbe az alábbi erőforrások: R (R1: 10; R2: 5; R3: 7)

A rendszerbe 5 processz van: P0, P1, P2, P3, P4

Kérdés: Kielégíthető-e P1 (1,0,2), P4 (3,3,0) ill. P0 (0,2,0) kérése úgy, hogy biztonságos legyen, holtpontmentesség szempontjából a rendszer - a következő kiinduló állapot alapján.

Külön-külön táblázatba oldja meg a feladatot!

- Határozza meg a processzek által igényelt erőforrások mátrixát?
- Határozza meg pillanatnyilag szabad erőforrások számát?
- Igazolja, magyarázza az egyes processzek végrehajtásának lehetséges sorrendjét - számolással?!”

Az összes osztály -erőforrások száma: (10, 5, 7)						
Kiinduló állapot						
	1. lépés			2. lépés		
	MAX. IGÉNY			FOGLAL		
	R1	R2	R3	R1	R2	R3
P0	7	5	3	0	1	0
P1	3	2	2	2	0	0
P2	9	0	2	3	0	2
P3	2	2	2	2	1	1
P4	4	3	3	0	0	2

Megoldás: alapesetben: legalább egy olyan sorrendet lehet találni, amivel minden le tud futni (kép). Ugyanez nem mondható el a plusz igények esetében, mivel ha rögtön mindegyiket hozzáadjuk, két erőforrásból helyből mínusz érték marad. Ha egyenként adjuk hozzá később(ha egyáltalán lehet ilyen), akkor sem megoldható, mivel max a P3-at lehet ezen esetekben lefuttatni, ha egyáltalán azt is, és utáni megint lebénul a rendszer.

Banker algoritmus

R1=10 R2=5 R3=7

①

Max igény

	R1	R2	R3
P0	7	5	3
P1	3	2	2
P2	6	0	2
P3	2	2	2
P4	4	3	3

Foglalt

	R1	R2	R3
P0	0	1	0
P1	2	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2
	4	2	5

Kéréslet

R1	R2	R3
3	3	2

P1 befut:

Max igény

	R1	R2	R3
P0	7	5	3
P1	0	0	0
P2	6	0	2
P3	2	2	2
P4	4	3	3

Foglalt

	R1	R2	R3
P0	0	1	0
P1	0	0	0
P2	3	0	2
P3	2	1	1
P4	0	0	2
	5	2	5

Kéréslet

R1	R2	R3
5	3	2

P3 befut:

Max igény

	R1	R2	R3
P0	7	5	3
P1	0	0	0
P2	6	0	2
P3	0	0	0
P4	4	3	3

Foglalt

	R1	R2	R3
P0	0	1	0
P1	0	0	0
P2	3	0	2
P3	0	0	0
P4	0	0	2
	3	1	4

Kéréslet

R1	R2	R3
7	4	3

→ P4 befut:

Max igény

	R1	R2	R3
P0	7	5	3
P1	0	0	0
P2	6	0	2

Foglalt

	R1	R2	R3
P0	0	1	0
P1	0	0	0
P2	3	0	2
P3	0	0	0
P4	0	0	0
	3	1	4

Kéréslet

R1	R2	R3
7	4	3

P2 befut:

R1	R2	R3
10	4	7

→ P0 befut és vége.

2. feladat:

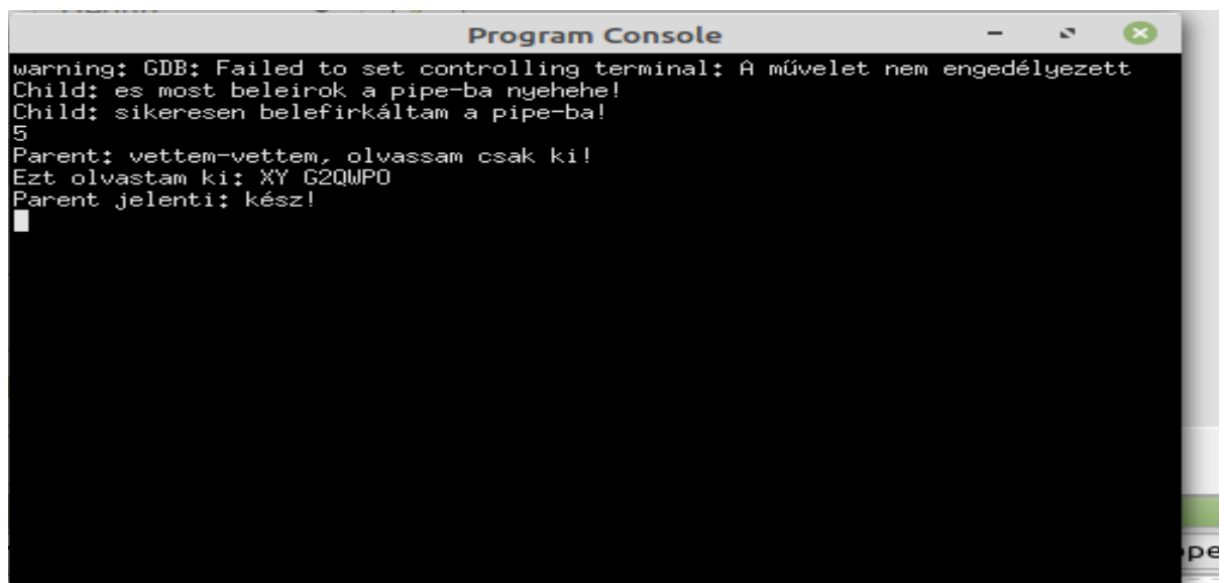
2. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetékét, a gyerek processz beleír egy szöveget a csővezetékbe (A kiírt szöveg: XY neptunkod), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.

Mentés: neptunkod_unnamed.c

Megoldás:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/wait.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  #define MSIZE 128
8  int main()
9  {
10     char inBuffer[MSIZE];
11     int p[2], nbytes, pid;
12     if(pipe(p)<0)
13     {
14         perror("Pipe hiba\n");
15         exit(1);
16     }
17     pid=fork();
18     if(pid<0)
19     {
20         exit(2);
21     }
22     if(pid==0)
23     {
24         printf("Child: es most beleírok a pipe-ba nyehehe!\n");
25         write(p[1], "XY G2QWPO", MSIZE);
26         printf("Child: sikeresen belefirkáltam a pipe-ba!\n");
27     }
28     else if(pid>0)
29     {
30         wait(NULL);
31         printf("Parent: vettem-vettem, olvassam csak ki!\n");
32         read(p[0], inBuffer, MSIZE);
33         printf("Ezt olvastam ki: %s\n", inBuffer);
34         printf("Parent jelenti: kész!\n");
35     }
```

Kimenet:

A screenshot of a 'Program Console' window. The window has a title bar with standard Windows window controls (minimize, maximize, close). The console output is as follows:

```
warning: GDB: Failed to set controlling terminal: A művelet nem engedélyezett
Child: es most beleírok a pipe-ba nyehehe!
Child: sikeresen belefirkáltam a pipe-ba!
5
Parent: vettem-vettem, olvassam csak ki!
Ezt olvastam ki: XY G2QWPO
Parent jelenti: kész!
```

A cursor is visible on the line following 'Parent jelenti: kész!'.

pe

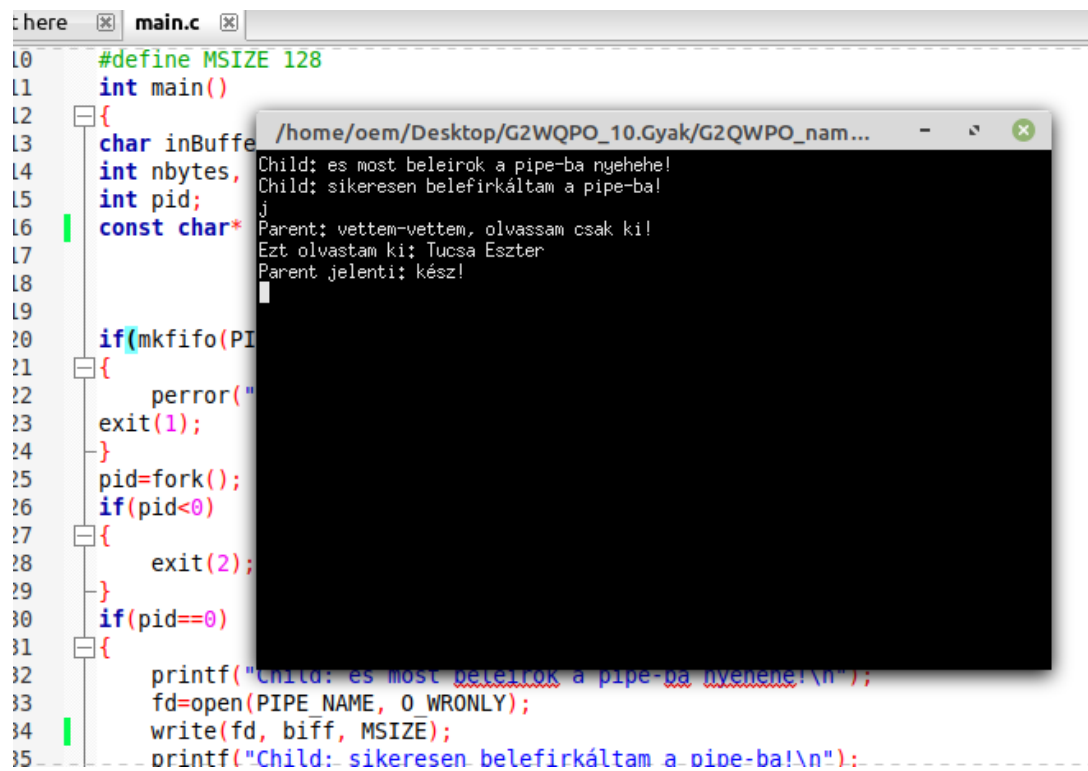
3. feladat:

3. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetékét (neve: neptunkod), a gyerek processz beleír egy szöveget a csővezetékbe (A hallgató neve: pl.: Keserű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.
Mentés: neptunkod_named.c

Megoldás:

```
5  #include <unistd.h>
6  #include <sys/stat.h>
7  #include <string.h>
8  #include <fcntl.h>
9  #define PIPE_NAME "G2QWPO"
10 #define MSIZE 128
11 int main()
12 {
13     char inBuffer[MSIZE];
14     int nbytes, fd;
15     int pid;
16     const char* biff="Tucsa Eszter";
17
18
19
20     if(mkfifo(PIPE_NAME,0666)<0)
21     {
22         perror("mkfifo\n");
23         exit(1);
24     }
25     pid=fork();
26     if(pid<0)
27     {
28         exit(2);
29     }
30     if(pid==0)
31     {
32         printf("Child: es most beleírok a pipe-ba nyehehe!\n");
33         fd=open(PIPE_NAME, O_WRONLY);
34         write(fd, biff, MSIZE);
35         printf("Child: sikeresen belefírkáltam a pipe-ba!\n");
36         close(fd);
37     }
38     else if(pid>0)
39     {
40         fd=open(PIPE_NAME, O_RDONLY);
41
42         wait(NULL);
43         printf("Parent: vettem-vettem, olvassam csak ki!\n");
44
45         read(fd, inBuffer, MSIZE);
46         close(fd);
47         printf("Ezt olvastam ki: %s\n", inBuffer);
48         printf("Parent jelenti: kész!\n");
49
50     }
51 }
```

Kimenete:



The image shows a code editor window with a file named `main.c` and a terminal window displaying the output of the program. The code in `main.c` defines a pipe, forks a child process, and uses `write` and `read` to communicate. The terminal output shows the child process writing a message to the pipe, the parent process reading it, and then the child process writing another message.

```
10 #define MSIZE 128
11 int main()
12 {
13     char inBuffer[MSIZE];
14     int nbytes;
15     int pid;
16     const char* pipeName = "/home/oem/Desktop/G2WQPO_10.Gyak/G2WQPO_namePipe";
17
18     if(mkfifo(pipeName, 0666))
19     {
20         perror("mkfifo failed");
21         exit(1);
22     }
23     pid=fork();
24     if(pid<0)
25     {
26         perror("fork failed");
27         exit(2);
28     }
29     if(pid==0)
30     {
31         printf("Child: es most beleírok a pipe-ba nyehehe!\n");
32         fd=open(pipeName, O_WRONLY);
33         write(fd, "nyehhehe", 8);
34         printf("Child: sikeresen belefirkáltam a pipe-ba!\n");
35     }
36     else
37     {
38         read(inBuffer, MSIZE);
39         printf("Parent: vettem-vettem, olvassam csak ki!\n");
40         printf("Ezt olvastam ki: %s\n", inBuffer);
41         printf("Parent jelenti: kész!\n");
42     }
43 }
```

Terminal Output:

```
Child: es most beleírok a pipe-ba nyehehe!
Child: sikeresen belefirkáltam a pipe-ba!
Parent: vettem-vettem, olvassam csak ki!
Ezt olvastam ki: Tucsá Eszter
Parent jelenti: kész!
```