Operációs rendszerek BSc

Féléves beadandó 2022. 03. 05.

Miskolc, 2022

Készítette:

Berki Viktor Bsc Porgramtervező Informatikus ZY5P7F

1. feladat - Parent-Child-Pipe-Wait C feladat

Létrehozzunk két file diszkruptort, azért mert a cső két végén kommunikál mind két fél.

Meghívjuk a Pipe(); függvényt és paraméternek írjuk a diszkruptort.

Létre hozzunk egy egész számos változót és eltároljúk a szülő pid-jét, úgy hogy létre hozzunk a gyermek process-t, a fork(); függvénnyel.

If statement-tel eldöntjük hogy a gyermek vagy a szülő process-szel dolgozunk.

Close(); függvényel bezárjuk az olvasó rész a szülőnél, majd write(); függvénnyel beleírjuk a csőbe a pidet. Majd bezárjuk az írás részét is.

A gyermek eltárolja egy változóban az olvasott adatot a read(); függvénnyel.

Majd bezárja az olvasó részt és elküldi a saját pid-jét a szülőnek és bezárja az írót is.

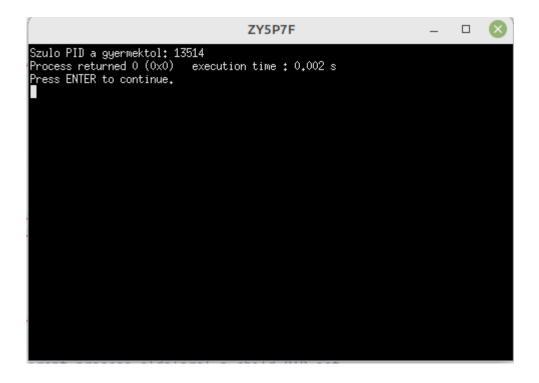
A szülő újra elolvassa a gyermek pidjét, majd a wait(); függvényel és a null paraméterrel megvárja hogy a gyermek meghaljon.

```
File Edit View Search Project Build Debug wxSmith Tools Tools+ Plugins DoxyBlocks Settings Valgrind Help
      1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <wait.h>>
                                                                                                                            int fd[2]; //Két file diskcriptor van lêtre hozzva. fd[0] az olvasó. fd[1] az iró része a csőnek
pipefdd;
int id = fork(); //Eltárolja a proccess és a child PID-iét, ha 0 akkor a child, ha más adott szám a parent proccess
if(id != 0)
                                                                                             10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
                                                                                                                                        close(fd[0]); //Beráriuk a parent groccess oldaláról az olvasó diskcriptort
write(fd[1],6d, sizeof(int)); //Bela [r]uk a csóbe a parent PID-jét
close(fd[1]); //Bezáriuk a parent proccess oldaláról ag irás diskcriptort
                                                                                                                                     int x; read(fd[0], 6x, sizeof(int)); //Balvassub a child pracces oldalarol a parent PID-jét printf("Spulo PID a quermektol: Md",x); write(fd[1],8x,sizeof(int)); //Balt iriub a ciobe a child PID-jét close(fd[0]); //Balt iriub a child praccess oldalarol az olvaso diskriptort close(fd[1]); //Bazairub a child praccess oldalarol az olvaso diskriptort close(fd[1]); //Bazairub a child praccess oldalarol az irab diskriptort
                                                                                                                            if(id != 0)
                                                                                                                                          wait(NULL); //Benyária mig a child proccess végez a wait(): metódus, maid a NULL paraméter miatt, pedig addig vária még meg nem hal
close(fd[1]); //Bezáriuk a parent proccess oldaláról az írás diskcriptort
int v:
                                                                                                                                            int y;
read(fd[0],&y,sizeof(int)); //Bolvassuk a parent procces oldaláról a child PID-jét
close(fd[0]); //Bezárjuk a parent proccess oldaláról az olvasó diskcriptort
                                                                                                                             return θ; // A <u>szülő proccess</u> meghal
                                                                                  Ŧ | P. Code:Blocks: 🔞 🔍 Search results: 🔞 | P. DoxyBlocks: 🔞 🐧 Debugger: 🔞 (P. DepCheck/Vera++ 🔘 💸 Build log: 🔞 | P. Opcheck/Vera++ messages: 🔞 📌 Build messages: 🔞 (P. Cccce): P. Cccope: 🔞 🐊 Thread search: 🙈 🐍 Closed files list: 😸 (P. DepCheck/Vera++ messages): P. Cccope: P. DepCheck/Vera++ messages: P. Cccce; P. DepCheck/Vera++ messages: P. De
                                                                                               Line Message == Build: Debug in 2YSPTF (compiler: ON GCC Compiler) == Freenriese/As... 4 warning: extra tokens at end of sinclude directive
                                                                                                                                                  warning: extra tokens at end of #include directive

— Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s))

— Run: Debug in ZYSP7F (compiler: GNU GCC Compiler) —
                                                                                                                                                                                                                                                                                                                                                                                                                                                  Unix (LF) UTF-8 Line 36, Col 41, Pos 1555 Insert
(m) 🛅 🗿 🛅 📑 🙆

② ① © □ ♣ ♬ ② 13:31
```



2.feladat

A Foglaltokat összeadjuk oszloponként, majd kivonjuk az alap erőforrásunkból, így megkapjuk az aktív szabad errőforásokat

Szükség kiszámításához minden egyes max. igény process memóriáját kivontuk a foglalt cellákból. Megkerestük a legkisebb errőforrást igénylőt a szükség processzok közül és összehasonlítottuk a szabad erőforrásokkal, ha nagyobb egyenlő volt a szabad erőforrás akkor a futási sorrendbe be került a process és hozzá adtuk a foglalt memóriáit a szabad erőforrásokhoz. Így végig, míg vissza nem kapjuk az alap erőforrásunkat.

				_																
	Kiind	uló állapot									R1:8	R2:9	R3:13							
	Foglal							Max. igény		Szab	ad Erőfori	rások		Szükség				Futási sorrend		
	R1	R2	R3				R1	R2	R3		R1	R2	R3						P3	
P1		2 2	2	3				4	2	5	3	3	3 5		2	0	1	2	P1	
P2		0 1	L	1				7	7	7	4	5	5 7		7	6		6	P4	
P3		1 2	2	2				1	4	3	6	7	7 10		0	2	:	1	P2	
P4		2 1	L	2				3	7	4	8	8	3 12		1	6		2		
		5 6	5	8							8	9	13							
P2: 1,2,2		A feladat				B feladat					C feladat									
Válasz: Teljesíthető		Biztonságos			P2 előtt:	3		4	6		P3									
					P2 után:	4	ı	6	8		P1									
											P4									
											P2									