JEGYZŐKÖNYV

Operációs rendszerek BSc

2022 Tavasz féléves feladat

Készítette:
Bordás Milán Bsc
Programtervező informatikus
WJB0DC

Miskolc, 2022

1.feladat leírása:

Adott négy processz (A, B, C, D) a rendszerbe, induláskor a p_cpu értéke A=0, B=0, C=0, D=0. A rendszerben a P_USER = 60.
Az óraütés 1 indul, a befejezés 301-ig. Induláskor a p_usrpri A=60, B=60, C=65 és D=60. Induláskor a p_nice értéke A=0, B=0, C=5 és D=0.
a.) Határozza meg az ütemezést RR nélkül 301 óraütésig - táblázatba!
b.) Minden óraütem esetén határozza meg a processzek sorrendjét óraütés előtt/után.
c.) Igazolja a számítással a tanultak alapján.

Elkészítésének lépései:

Mindig a legkisebb prioritású processz fut.

Minden őraütésnél a futó processz p_cpu ideje 1-gyel nő.
Minden 100.dik óraütésnél a futó processz p_cpu idejét és p_pri-jét újraszámoljuk
p_cpu = p_cpu/2

p_pri = p_user+p_cpu/4+p_nice*2

Megoldás:

	A folyamat		B folyamat		C folyamat		D folyamat		Átütemezés			
Óraütés	p_pri	p_cpu	p_pri	p_cpu	p_pri	p_cpu	p_pri	p_cpu	elötte	utána	P_USER:	60
kiindulás	60	0	60	0	65	0	60	0			A nice:	0
1	60	1	60	0	65	0	60	0	A	A	B nice:	0
2	60	2	60	0	65	0	60	0	A	A	C nice:	5
3	60	3	60	0	65	0	60	0	A	A	D nice:	0
											konst1:	4
9	60	9	60	0	65	0	60	0	A	A	konst2:	2
10	60	10	60	0	65	0	60	0	A	A		
11	60	11	60	0	65	0	60	0	A	A		
99	60	99	60	0	65	0	60	0	A	A		
100	73	50	60	0	70	0	60	0	A	B		
101	73	50	60	1	70	0	60	0	B	B		
110	73	50	60	10	70	0	60	0	B	B		
111	73	50	60	11	70	0	60	0	B	B		
112	73	50	60	12	70	0	60	0	B	B		
									B	B		
199	73	50	60	99	70	0	60	0	B	B		
200	66	25	73	50	70	0	60	0	B	D		
201	66	25	73	50	70	0	60	1	D	D		
210	66	25	73	50	70	0	60	10	D	D		
211	66	25	73	50	70	0	60	11	D	D		
212	66	25	73	50	70	0	60	12	D	D		
299	66	25	73	50	70	0	60	99	D	D		
300	63	13	66	25	70	0	73	50	D	A		
301	63	14	66	25	70	0	73	50	A	A		

2.feladat leírása:

Keszitsen egy programot, ami egy 1000 elemu egesz szam tipusu tombben ugy keresi meg a maximumot, hogy 10 szalat futtat parhuzamosan, amik kozul mindegyik 100 elemet vizsgal meg. Az eredeti szulo processz nem szamol, viszont o gyujti be az eredmenyeket, amit pipe-on keresztul var a processzektol.

Elkészítésének lépései:

Létrehozunk egy int típusú 1000 elemű tömböt, feltöltjük random egészekkel (1-9900-ig).
A szülő processzben létrehozunk két pipeot, az elsőbe betöltjük az 1000 elemet, a másodikban majd a gyerek processzek adják vissza az eredményeket.
Létrehozzuk egy for ciklusban a gyerekeket, mindegyik gyerek processzel kiolvastatunk 100 elemet, majd elvégeztetjük velük a maximumkeresést.
Visszaküldjük a pipeon keresztül az eredményt, majd kilépünk a gyerekből.
A szülő processzben összegyűjtjük az eredményeket, és ezen 10 számra is elvégzünk egy maximumkeresést, majd kiírjuk az eredményt.

Megoldás:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
void feltolt(int* tomb, int length); //feltoltjuk az 1000 elemu tombot
int maxkeres(int* tomb, int length); //maxkereses 100 es 10 re
int main()
{
    int db = 1000;
    int processDb = 10;
    int array[db];
    feltolt(array, db);
    int i;
    char str[4]; //pipeba ezen keresztul mennek az adatok
    int p[2]; //pipe amibol a gyerekek olvasnak
    int p2[2]; //pipe amibol a szulo olvas (1-el is lehet mukodik)
    if(pipe(p)<0)
    {
        exit(1);
    }
    if(pipe(p2)<0) //pipeok letrehozasa, ha nem sikerul kilep
    {
        exit(1);
    }
    for(i = 0; i< db; i++)
    {
        sprintf(str,"%d",array[i]); //pipeba stringkent toltjuk az elemeket, majd konvertaljuk
        write(p[1],str,4);
    }
    close(p[1]);
    for(i = 0; i < processDb; i++)
    {
        if(fork() == 0) //itt letrehozunk egy gyereket ciklusonkent, majd annak megadjuk a feladatot
        {
            int szamok[100];
            int j;
            char be[4]; //1. pipebol ide jonnek a szamok
            char vissza[4]; //2. pipeba innen irunk
            for(j = 0;j<100;j++)
            {
                read(p[0],be,4);
                szamok[j] = atoi(be); //konvertaljuk a stringet intbe
            }
            close(p[0]);
            int max = maxkeres(szamok,100);
            sprintf(vissza,"%d",max);
            write(p2[1],vissza,4); //visszaadjuk a 2.pipeba a maximumat a darabolt tombnek
            close(p2[1]);
            exit(0); //kilepunk a gyerekbol, tehat csak a fork() == 0-tol idaigi reszt hajtja vegre
            }
    }
    for(i = 0; i < processDb; i++)
    {
        wait(NULL); //megvarunk minden processt
```

```
        wait(NULL); //megvarunk minden processt
    }
    int maxok[processDb];
    for(i = 0; i< processDb;i++)
    {
        read(p2[0],str,4);
        maxok[i] = atoi(str); //stringkent jottek a maxok is
    }
    printf("A tomb max erteke: %d\n",maxkeres(maxok,processDb));
    return 0;
}
//fuggvenyek deklaracioi
void feltolt(int* tomb, int length)
{
    srand(time(0));
    int i;
    for(i = 0; i < length; i++)
    {
        tomb[i] = rand()%9900+1;
    }
}
int maxkeres(int* tomb, int length)
{
    int i;
    int max = tomb[0];
    for(i = 0; i< length; i++)
    {
        if(max<tomb[i])
        {
            max = tomb[i];
        }
    }
    return max;
}
```

A futások eredménye (11db, 1-9900ig töltöttük a tömböt):

```
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9887
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9882
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9874
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9874
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9874
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9895
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9900
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9887
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9896
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9896
bordas7@jerry:~/OS/beadando$ ./tizprocess.c
A tomb max erteke: 9878
bordas7@jerry:~/OS/beadando$
```