Operációs rendszerek BSc

7.gyak.

2021. 03. 29.

Készítette:
Ábrahám Péter István

Gazdaságinformatikus
Neptunkód: **A4XIOV**

Miskolc, 2021

Feladatok

1. Adott négy processz a rendszerben, melynek beérkezési sorrendje: A, B, C és D. Minden processz USER módban fut és mindegyik processz futásra kész. `P_USER = 60`

Kezdetben mindegyik processz `p_uspri = 60`.

Az A, B, C processz `p_nice = 0`, a D processz `p_nice = 5`.

Mindegyik processz `p_cpu = 0`, az óraütés 1 indul, a befejezés legyen 201. óraütés-ig.

a.) Határozza meg az ütemezést RR nélkül és az ütemezést RR-nal - külön-külön táblázatba.

b.) Minden óraütem esetén határozza meg a processzek sorrendjét óraütés előtt/után.

c.) Igazolja a számítással a tanultak alapján.

$KF = 2 \cdot FK/(2 \cdot FK+1)$ - korrekciós faktor;

p_cpu = p_cpu * KF, ahol KF értéke 1/2;

p_pri = P_USER + p_cpu / 4 + 2 * p_nice;

A táblázat javasolt formája RR/RR nélkül a következő:

	A process		B process		C process		D process		Reschedule	
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_uspri	p_uspri	p_cpu	running before	running after
Starting point	60	0	60	0	60	0	60	0		
1	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

RR NÉLKÜL	A process		B process		C process		D process		Reschedule	
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	running before	running after
Starting point	60	0	60	0	60	0	60	0		
1	60	1	60	0	60	0	60	0		A
...										
99	60	99	60	0	60	0	60	0	A	A
100	60+50/4 73	100/2 50	60	0	60	0	60	0	A	B
101	73	50	60	1	60	0	60	0	B	B
...										
199	73	50	50	99	60	0	60	0	B	B
200	60+25/4 67	50/2 25	60+50/4 73	100/2 50	60	0	60	0	B	C
201	67	25	73	50	60	1	60	0	C	C
...										
299	67	25	73	50	60	99	60	0	C	C
300	60+13/4 63	25/2 13	60+25/4 67	50/2 25	60+50/4 73	100/2 50	60	0	C	D
301	63	13	67	25	73	50	60	1	D	D
...										
399	63	13	77	25	83	50	60	99	D	D
400	60+7/4 62	13/2 7	60+13/4 63	25/2 13	60+25/4 67	50/2 25	60+50/4+2*5 83	100/2 50	D	

RR	A process		B process		C process		D process		Reschedule	
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	running before	running after
Starting point	60	0	60	0	60	0	60	0		
1	60	1	60	0	60	0	60	0		A
...										
9	60	9	60	0	60	0	60	0	A	A
10	60	10	60	0	60	0	60	0	A	B
11	60	10	60	1	60	0	60	0	B	B
...	...	...	...	...	...	...	...	...	...	...
19	60	10	60	9	60	0	60	0	B	B
20	60	10	60	10	60	0	60	0	B	C
21	60	10	60	10	60	1	60	0	C	C
...	...	...	...	...	...	...	...	...	...	...
29	60	10	60	10	60	9	60	0	C	C
30	60	10	60	10	60	10	60	0	C	D
31	60	10	60	10	60	10	60	1	D	D
...	...	...	...	...	...	...	...	...	...	...
40	60	10	60	10	60	10	60	10	D	A
...	...	...	...	...	...	...	...	...	...	...
50	60	20	60	10	60	10	60	10	A	B
...	...	...	...	...	...	...	...	...	...	...
60	60	20	60	20	60	10	60	10	B	C
...	...	...	...	...	...	...	...	...	...	...
70	60	20	60	20	60	20	60	10	C	D
...	...	...	...	...	...	...	...	...	...	...
80	60	20	60	20	60	20	60	20	D	A
...	...	...	...	...	...	...	...	...	...	...
90	60	30	60	20	60	20	60	20	A	B
...	...	...	...	...	...	...	...	...	...	...
100	63	15	63	15	62	10	72	10	B	C
...	...	...	...	...	...	...	...	...	...	...
199	63	15	63	15	62	109	72	10		
200	61	7	61	7	73	55	71	5	C	A
201	61	8	61	7	73	55	71	5		

2. A tanult rendszerhívásokkal (`open()`, `read()/write()`, `close()` – ezek fogják a rendszerhívásokat tovább hívni) írjanak egy `neptunkod_openclose.c` programot, amely megnyit egy fájlt – `neptunkod.txt` (O_RDWR megnyitási móddal), tartalma: hallgató neve, szak , neptunkod.

A program következő műveleteket végezze:

- olvassa be a `neptunkod.txt` fájlt,
- hiba ellenőrzést,
- read() - kiolvassa a `neptunkod.txt` tartalmát és mennyit olvasott ki (byte), és kiírja konzolra.

- lseek() – pozícionálja a fájl kurzor helyét, ez legyen a fájl eleje: SEEK_SET, és kiírja a konzolra.
- write() - mennyit ír ki a konzolra.

Írjanak magyarázatot a jegyzőkönyvbe.

További információk: man 2 open; man 2 read; man 2 write; man 2 close.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(){
        int opener;
        char  *arr[100];
        int reader;

        if((opener = open("a4xiov.txt", O_RDWR)) < 0){
                perror("Open error.");
                return -1;
        }

        if((reader = read(opener, arr, sizeof(arr))) != 0){
                if(reader == -1){
                        perror("Read error.");
                        return -1;
                }
                int i;
                for(i=0; i<sizeof(arr) && arr[i] != NULL; i++);
                arr[i] = '\0'; // Lezárjuk a sort;
                printf("Number of bytes: %d, \nThe file itself contains: %s\n", reader, arr);
        }

        if(lseek(opener, 0, SEEK_SET) < 0){
                perror("Lseek error");
                return -1;
        }

        if((reader = write(opener, "Hello there\n", 11)) != 11){
                perror("Write error");
                return -1;
        }

        close(opener);
        return 0;
}
```