Operációs Rendszerek BSc

7. gyak.

2021. 03. 24.

Készítette:

Nyíri Beáta
Programtervező Informatikus
I40FDC

Miskolc, 2021

1. Adott négy processz a rendszerbe, melynek beérkezési sorrendje: A, B, C és D. Minden processz USER módban fut és mindegyik processz futásra kész.

   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·FK/2·FK+1 - korrekciós faktor;
   p_cpu = p_cpu * KF, ahol KF értéke 1/2;
   p_pri = P_USER + p_cpu / 4 + 2 * p_nice;

   RR nélkül:

	A process		B process		C process		D process		Reschelude	
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
...	...	...	...	...	60	...	...	...	A	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
...	...	...	...	...	60	...	...	...	B	B
199	73	50	60	99	60	0	60	0	B	B
200	73+25/4 80	50/2 25	60+50/4 73	100/2 50	60	0	60	0	B	

RR-val:

RR: 10ms	A process		B process		C process		D process		Reschelude	
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		A
1	60	1	60	0	60	0	60	0	A	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
...	...	...	...	...	...	...	...	...	...	...
20	60	10	60	10	60	0	60	0	B	C
...	...	...	...	...	...	...	...	...	...	...
30	60	10	60	10	60	10	60	0	C	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	A
...	...	...	...	...	...	...	...	...	...	...
99	60	39	60	20	60	20	60	20	A	A
100	50+20/4 55	40*0,5 20	50+10/4 53	20*0,5 10	50+10/4 53	20*0,5 10	50+10/4+2*5 63	20*0,5 10	A	B
101										
...										
110										
...										
120										
...										
130										
...										
140										
...										
150										
...										
160										
...										
170										
...										
180										
...										
190										
...										
199										
200										

?

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

   A program következő műveleteket végezze:
   - olvassa be a neptunkod.txt fájlt, melynek attribútuma: O_RDWR
   - hiba ellenőrzést,
   - write() - mennyit ír ki a konzolra.
   - 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.

   Együtt készített kód:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/file.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

int main()
{
    int fd, ret;
    char buf[32];

    buf[0]=0;

    fd=open("i40fdc.txt",O_RDWR);

    if(fd == -1){
        perror("open() hiba!");
        exit(-1);
    }

    ret=read(fd,buf,32);
    printf("read() olvasott %d bytot, ami a következő: %s\n",ret,buf);
    strcpy(buf,"I40FDC");

    ret=lseek(fd,0,SEEK_SET);
    printf("lseek() mondja: %d\n",ret);

    ret=write(fd,buf,6);
    printf("white() mondja: %d\n",ret);

    close(fd);

}
```

```
read() olvasott 7 bytot, ami a következő: I40FDC

lseek() mondja: 0
white() mondja: 6

Process returned 0 (0x0)   execution time : 0.008 s
Press ENTER to continue.
```

Mi történik, ha felcseréljük a hívások sorrendjét?

1. write(), 2. read(), 3. lseek()

```
white() mondja: 6
read() olvasott 1 bytot, ami a következő:
@@
lseek() mondja: 0

Process returned 0 (0x0)   execution time : 0.013 s
Press ENTER to continue.
```

A write() még működik, de a read() már nem jót olvas ki és 1 byte-ot ad vissza.

1. lseek(), 2. write(), 3. read()

```
lseek() mondja: 0
write() mondja: 6
read() olvasott 1 bytot, ami a következő:
@@B

Process returned 0 (0x0)   execution time : 0.005 s
Press ENTER to continue.
```

A read() itt is csak 1 byteot olvas ki.

1. write(), 2. sleek(), 3. read()

```
write() mondja: 6
lseek() mondja: 0
read() olvasott 7 bytot, ami a következő:

Process returned 0 (0x0)   execution time : 0.016 s
Press ENTER to continue.
```

A write() és a sleek() is működik, a read() kiolvassa a byteokat, viszont a „buf”-ot nem írja ki.