Klasszikus UNIX ütemezés (RR/RR nélkül)

(EXCEL táblázat is megtalálható a repo-ban)

Feladat:

Adott négy processz a rendszerbe, melynek beérkezési sorrendje: A, B, C és D. Minden processz USER módban fut és mindegyik processz futásra kész. **(P_USER- nek 50-et használtam) :**

Round Robin:

RR	A		B		C		D		Rescedule	
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	unning befor	unning after
Starting poir	60	0	60	0	60	0	60	0		A
1	60	1	60	0	60	0	60	0	A	A
2	60	2	60	0	60	0	60	0	A	A
3	60	3	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
19	60	10	60	9	60	0	60	0	B	B
20	60	10	60	10	60	0	60	0	B	C
29	60	10	60	10	60	9	60	0	C	C
30	60	10	60	10	60	10	60	0	C	D
39	60	10	60	10	60	9	60	9	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	A	B
70	60	20	60	20	60	20	60	10	B	C
80	60	20	60	20	60	20	60	20	C	D
90	60	30	60	20	60	20	60	20	D	A
99	60	39	60	20	60	20	60	20	A	A
100	55	20	52.5	10	52.5	10	62.5	10	A	B
110	55	20	52.5	20	52.5	20	62.5	10	B	C
120	55	30	52.5	20	52.5	20	62.5	10	C	A
130	55	40	52.5	20	52.5	20	62.5	20	A	D
140	55	40	52.5	30	52.5	20	62.5	20	D	B
150	55	40	52.5	30	52.5	30	62.5	20	B	C
160	55	50	52.5	30	52.5	30	62.5	20	C	A
170	55	50	52.5	30	52.5	30	62.5	30	A	D
180	55	50	52.5	40	52.5	30	62.5	30	D	B
190	55	50	52.5	40	52.5	40	62.5	30	B	C
199	55	50	52.5	40	52.5	49	62.5	30	C	C
200	56.25	25	55	20	56.25	25	63.75	15	C	B
201	56.25	25	55	21	56.25	25	63.75	15	B	B

First Come First Served (itt is a P_USER = 50):

FCFS	A		B		C		D		Rescedule	
Clock tick	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	p_uspri	p_cpu	running before	running after
1	60	0	60	0	60	0	60	0		A
2	60	1	60	0	60	0	60	0	A	A
3	60	2	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	A
99	60	99	60	0	60	0	60	0	A	A
100	62.5	50	50	0	50	0	50	0	A	B
199	62.5	50	50	99	50	0	50	0	B	B
200	56.25	25	62.5	50	50	0	50	0	B	C
201	56.25	25	62.5	50	50	1	50	0	C	C

2. feladat:

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.

open():

```
descriptor = open("KLYSO1.txt",O_RDWR);
if (descriptor == -1){
    perror("Hiba a megnyitásnál:");
    exit(descriptor);
}
printf("Descrpitor: %d\n", descriptor);
```

open függvénnyel megpróbáljuk megnyitni a text fájlunkat, a visszatérési értéket a descriptor tartalmazza.

Később hibakeresésnél a descpritor értékét vizsgáljuk, ha

-1 akkor valami hiba történt a beolvasásánál, ezért kiírunk egy hibaüzenetet.

read()

```
dread = read(descriptor, buffer, 15);
if (dread == -1){
    perror("Hiba az olvasásánál:");
    exit(dseek);
}
printf("Read: %d\n", dread);
printf("Ertek: %s", buffer);
```

read() függvénnyel megpróbáljuk beolvasni a text fájlunk tartalmát descriptor értékét a dread változóban tároljuk amit megvizsgálunk, és ha értéke -1 akkor hiba történt és kiírunk egy hibaüzenetet.

lseek()

```
dseek = lseek(descriptor, 0, SEEK_SET);
if (dseek == -1){
    perror("Hiba a kurzor pozícionálásnál:");
    exit(dseek);
}
printf("Pozíció: %d\n", dseek);
```

Kurzort pozícionáljuk a SEEK_SET-el a fájl elejére, ha hiba van akkor hibaüzenetet írunk ki. Aztán kiírjuk a kurzor helyét.

write()

```
strcpy(buffer, "KLYSO1");
bLength = strlen(buffer);
dwrite = write(descriptor, buffer, bLength);
if (dwrite == -1){
    perror("Hiba az írásnál:");
    exit(dwrite);
}
printf("Hossz(byte): %d", dwrite);
```

write függvénnyel megpróbálunk beírni a „KLYSO1”-textet. Ha hiba van akkor hibaüzenetet írunk.