Operációs rendszerek BSc

3.Gyakorlat

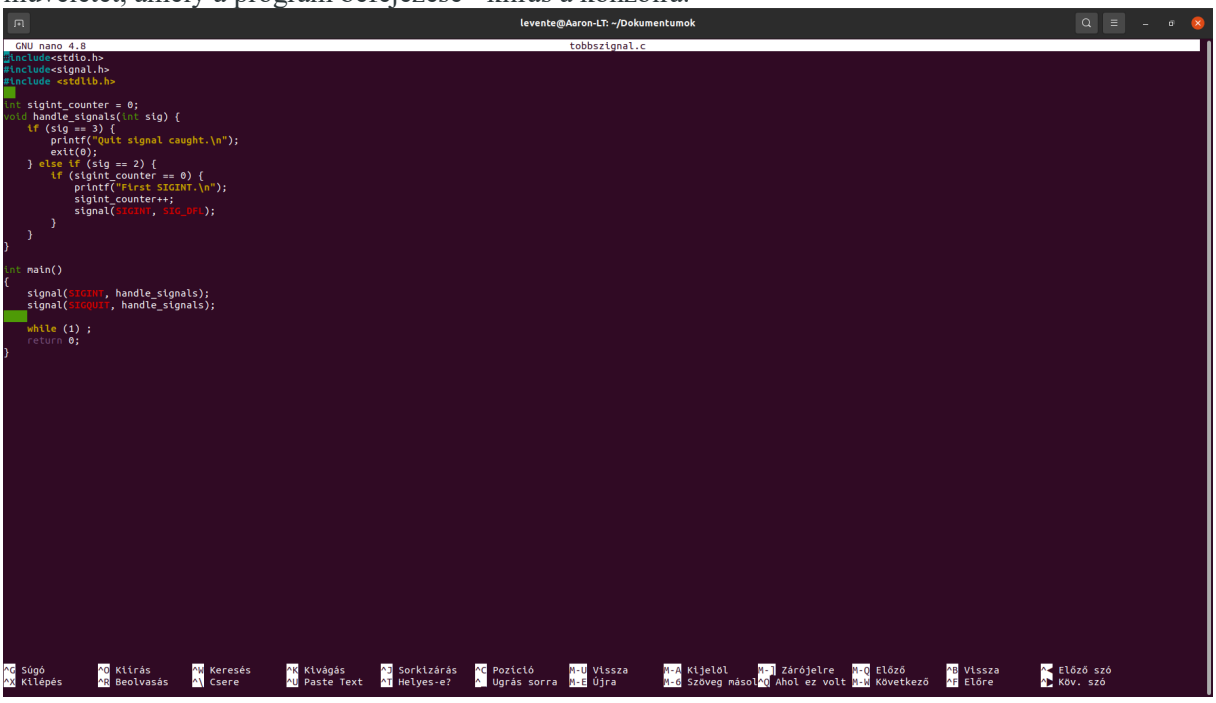
Készítette: **Szabó Levente**
Neptunkód: **GF2465**

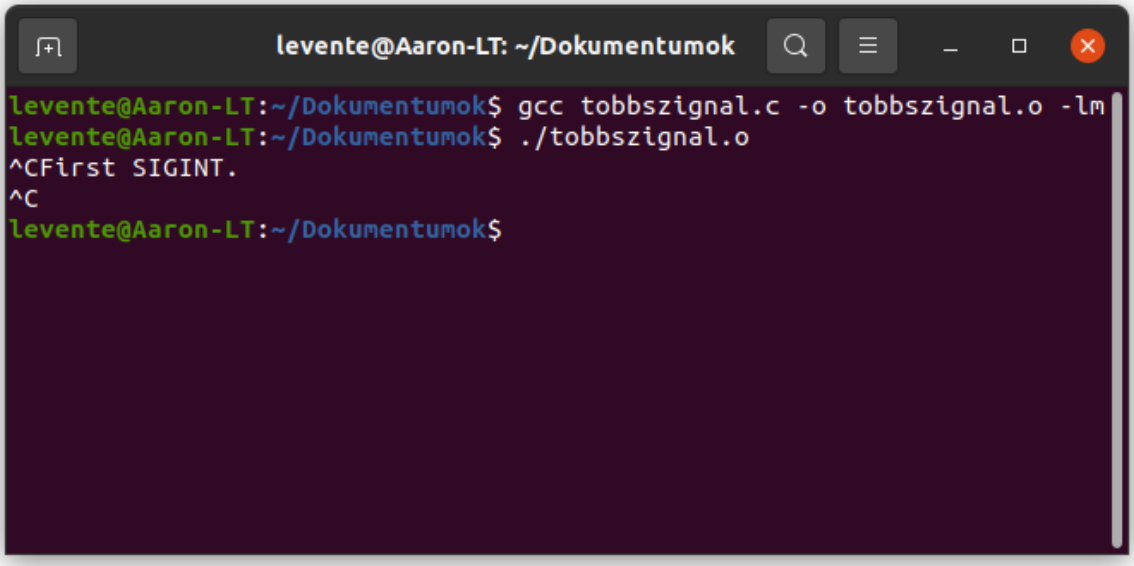
1. Adott négy processz a rendszerbe, melynek beérkezési sorrendje: A, B, C és D. Minden processz USER módban fut és mindegyik processz futásra kész.
Kezdetben mindegyik processz `p_uspri = 60`.
Az A, B, C processz `p_nice = 0`, a D processz `p_nice = 5`.
Mindegyik processz `p_cpu = 0`, az óraütés 1 indul, a befejezés legyen 201. óraütés-ig.

RR-nal	A		B		C		D			
óraütés	**usr_pri**	**p_cpu**	**usr_pri**	**p_cpu**	**usr_pri**	**p_cpu**	**usr_pri**	**p_cpu**	**Előző proc**	**Következő proc**
0	60	0	60	0	60	0	60	0		A
10	60	10	60	0	60	0	60	0	A	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	B
100	66	25	66	25	64	17	74	17	C	D
110	66	25	66	25	64	27	74	17	C	A
120	66	35	66	25	64	27	74	17	A	B
130	66	35	66	35	64	27	74	17	B	D
140	66	35	66	35	64	27	74	27	D	C
150	66	35	66	35	64	37	74	27	C	A
160	66	45	66	35	64	37	74	27	A	B
170	66	45	66	45	64	37	74	27	B	D
180	66	45	66	45	64	37	74	37	D	C
190	66	45	66	45	64	47	74	37	C	A
200	78	47	76	39	74	39	91	31	A	B

A/B/C/D process nice = 5
p_pri=P_USER + p_cpu/2 + 2*p_nice
p_cpu=p_cpu/2

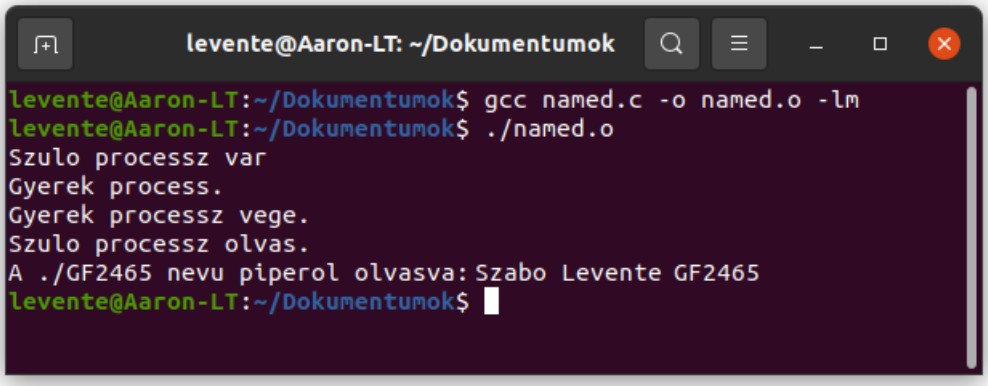
2. Készítse el a következő feladatot, melyben egy szignálkezelő több szignált is tud kezelni
a.) Készítsen egy szignál kezelőt (handleSignals), amely a SIGINT (CTRL + C) vagy SIGQUIT (CTRL + \) jelek fogására vagy kezelésére képes.
b.) Ha a felhasználó SIGQUIT jelet generál (akár kill paranccsal, akár billentyűzetről a CTRL + \) a kezelő egyszerűen kiírja az üzenetet visszatérési értékét – a konzolra.
c.) Ha a felhasználó először generálja a SIGINT jelet (akár kill paranccsal, akár billentyűzetről a CTRL + C), akkor a jelet úgy módosítja, hogy a következő alkalommal alapértelmezett műveletet hajtson végre (a SIG_DFL) – kiírás a konzolra.
d.) Ha a felhasználó másodszor generálja a SIGINT jelet, akkor végrehajt egy alapértelmezett műveletet, amely a program befejezése - kiírás a konzolra.





3. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy csővezetéket, a gyerek processz beleír egy szöveget a csővezetékbe (A kiírt szöveg: XY neptunkod), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.

4. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetéket (neve: neptunkod), a gyerek processz beleír egy szöveget a csővezetékbe (A hallgató neve:pl. Keserű Ottó), a szülő processz ezt kiolvassa, és kiírja a standard kimenetre.



5. Adott egy rendszerbe az összes **osztály-erőforrások száma**: R (R1: 10; R2: 9; R3: 12)
A rendszerbe 4 processz van: P1, P2, P3, P4.
Biztonságos-e holtpontmentesség szempontjából a rendszer - a következő *kiinduló állapot* alapján?
a) Határozza meg a folyamatok *által igényelt erőforrások mátrixát*?
b) Határozza meg *pillanatnyilag szabad erőforrások számát*?
c) Igazolja az egyes *processzek* végrehajtásának *lehetséges sorrendjét - számolással*?

Maximális igény

	R1	R2	R3
P1	4	4	5
P2	1	4	3
P3	6	7	7
P4	3	7	10

Foglalási igény

	R1	R2	R3
P1	2	2	3
P2	1	2	2
P3	0	1	3
P4	2	1	2

a)

Erőforrás mátrix

	R1	R2	R3
P1	2	2	2
P2	0	2	1
P3	6	6	4
P4	1	6	8

b)

Szabad erőforrások száma:												
Max_r	10	9	12	-	5	6	10	=	5	3	2	

c)

Foglalási igény

	R1	R2	R3
P1	2	2	3
P2	1	2	2
P3	0	1	3
P4	2	1	2

Maximális igény

	R1	R2	R3
P1	4	4	5
P2	1	4	3
P3	6	7	7
P4	3	7	10

Várható igény

	R1	R2	R3
P1	2	2	2
P2	0	2	1
P3	6	6	4
P4	1	6	8

Induló készlet: (5,3,2) ez P1, vagy P2 igényére elég
Ha P1-et választjuk: (5, 3, 2) - (2, 2, 2) + (2, 2, 2) + (2, 2, 3) = (5, 3, 2) + (2, 2, 3) = (7, 5, 5)
Ha P2-t választjuk: (7, 5, 5) - (0, 2, 1) + (1, 2, 2) = (7, 5, 5) + (1, 2, 2) = (8, 7, 7)