

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 óráütés 1 indul, a befejezés legyen 201. óráütés-ig.

RR-nal	A		B		C		D		Előző proc	Következő proc										
óraütés	usr_pri	p_cpu	usr_pri	p_cpu	usr_pri	p_cpu	usr_pri	p_cpu												
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.

```

levente@Aaron-LT: ~/Dokumentumok
GNU nano 4.8 tobbszignalt.c
#include<stdio.h>
#include<signal.h>
#include<stdlib.h>

int sigint_counter = 0;
void handle_signals(int sig) {
    if (sig == 3) {
        printf("quit signal caught.\n");
        exit(0);
    } else if (sig == 2) {
        if (sigint_counter == 0) {
            printf("First SIGINT.\n");
            sigint_counter++;
            signal(SIGINT, sig_DFL);
        }
    }
}

int main()
{
    signal(SIGINT, handle_signals);
    signal(SIGQUIT, handle_signals);

    while (1) ;
    return 0;
}

```

```
levente@Aaron-LT: ~/Dokumentumok
levente@Aaron-LT:~/Dokumentumok$ gcc tobbszignal.c -o tobbszignal.o -lm
levente@Aaron-LT:~/Dokumentumok$ ./tobbszignal.o
^CFirst SIGINT.
^C
levente@Aaron-LT:~/Dokumentumok$
```

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

```
GNU nano 4.8 unnaned.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main()
{
    int pipe1[2];
    pid_t p;

    if (pipe(pipe1) == -1) {
        fprintf(stderr, "Pipe1 hiba");
        return 1;
    }

    p = fork();

    if (p < 0) {
        fprintf(stderr, "fork hiba");
        return 1;
    }

    // Parent process
    else if (p > 0) {
        char str[100];

        printf("Szulo processz vár\n");

        wait(NULL);

        printf("Szulo process olvas.\n");

        read(pipe1[0], str, 100);
        printf("Pipelinnel olvasva: %s\n", str);
        close(pipe1[0]);
    }
    else {
        printf("Gyerek processz.\n");
        char output_string[100];
        strcpy(output_string, "Xyho levente GF2465");
        write(pipe1[1], output_string, strlen(output_string) + 1);
        close(pipe1[1]);

        exit(0);
    }
}
```

```
levente@Aaron-LT: ~/Dokumentumok
levente@Aaron-LT:~/Dokumentumok$ gcc unnamed.c -o unnamed.o -lm
levente@Aaron-LT:~/Dokumentumok$ ./unnamed.o
Szulo processz vár
Gyerek processz.
Szulo process olvas.
Pipelinerol olvasva: Szabo Levente GF2465
levente@Aaron-LT:~/Dokumentumok$
```

4. Készítsen C nyelvű programot, ahol egy szülő processz létrehoz egy nevesített csővezetékét (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.

```
GNU nano 4.0 named.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main()
{
    char* ffoname = "./GF2465";
    mkfifo(ffoname, 0666);

    int pipe;
    pid_t p;

    p = fork();

    if (p < 0) {
        fprintf(stderr, "fork hiba");
        return 1;
    } else if (p > 0) {
        char str[80];

        printf("Szulo processz var\n");

        wait(NULL);

        printf("Szulo processz olvas.\n");

        pipe = open(ffoname, O_RDONLY);
        read(pipe, str, 80);
        close(pipe);

        printf("A %s nevu piperol olvasva: %s\n", ffoname, str);
    } else {
        printf("Gyerek process.\n");

        char output_string[80];
        strcpy(output_string, "Szabo Levente GF2465\n");

        pipe = open(ffoname, O_WRONLY);
        write(pipe, output_string, strlen(output_string));
        close(pipe);

        printf("Gyerek processz vege.\n");
    }
}
```

```
levente@Aaron-LT: ~/Dokumentumok
levente@Aaron-LT:~/Dokumentumok$ gcc named.c -o named.o -lm
levente@Aaron-LT:~/Dokumentumok$ ./named.o
Szulo processz var
Gyerek process.
Gyerek processz vege.
Szulo processz olvas.
A ./GF2465 nevu piperol olvasva: Szabo Levente GF2465
levente@Aaron-LT:~/Dokumentumok$
```

Biztonságos-e holtpontmentesség szempontjából a rendszer - a következő *kiinduló állapot* alapján?

c) Igazolja az egyes *processzek* végrehajtásának *lehetséges sorrendjét* - számolással?

Maximális igény					Foglalási igény					Erőforrás mátrix				
R1	R2	R3			R1	R2	R3			R1	R2	R3		
P1	4	4	5		P1	2	2	3		P1	2	2	2	
P2	1	4	3		P2	1	2	2		P2	0	2	1	
P3	6	7	7		P3	0	1	3		P3	6	6	4	
P4	3	7	10		P4	2	1	2		P4	1	6	8	

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

Foglalási igény					Maximális igény					Várható igény				
R1	R2	R3			R1	R2	R3			R1	R2	R3		
P1	2	2	3		P1	4	4	5		P1	2	2	2	
P2	1	2	2		P2	1	4	3		P2	0	2	1	
P3	0	1	3		P3	6	7	7		P3	6	6	4	
P4	2	1	2		P4	3	7	10		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, 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)