

Operációs rendszerek BSc

3. Gyak.

2022. 04. 29.

Készítette:

Nagy Gergely Bsc

Mérnökinformatikus

G13RFP

Miskolc, 2022

UNIX processz ütemezés, szignálkezelés, holtpont

Jegyzőkönyv neve: *Neptunkod Gyak3.pdf és a* forrás file-k.

Irodalom:

Vincze Dávid: Operációs rendszerek - diasort.

Benyó Balázs, Fék Márk, Kiss István, Kóczy Annamária, Kondorosi Károly, Mészáros Tamás, Román Gyula, Szeberényi Imre, és Sziray, József: Operációs rendszerek mérnöki megközelítésben, Panem Kiadó, 2000, jegyzet/diák.

Szintén tanulmányozzák az előadáson bemutatott/kivetített mintapéldát és az URL linkhez tartozó irodalmat, majd oldják meg a feladatot.

„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 $p_{\text{nice}} = 0$, a D processz $p_{\text{nice}} = 5$.

Mindegyik processz $p_{cpu} = 0$, az óraütés 1 indul, a befejezés legyen 201. óraütés-ig.

- Határozza meg az ütemezést RR nélkül és az ütemezést RR-nal - külön-külön táblázatba.
- Minden órátem esetén határozza meg a processzek sorrendjét óráítés előtt/után.
- Igazolja a számítással a tanultak alapján.

A táblázat javasolt formája RR/RR nélkül a következő:

[illegible]

Starting point	A process		B process		C process		D process		running process	
	p_usrpri	p_cpu	p_usrpri	p_cpu	p_usrpri	p_cpu	p_usrpri	p_cpu	before	after
1	60	0	60	0	60	0	60	0		A
2	75	30	60	0	60	0	60	0	A	B
3	67	15	75	30	60	0	60	0	B	C
4	63	7	67	15	75	30	60	0	C	D
5	61	3	63	7	67	15	80	40	D	A
6	75	31	61	3	63	7	70	20	D	B
7	67	15	75	31	61	3	65	10	B	C
8	63	7	67	15	75	31	62	5	C	D

A process p_nice = 0
 B process p_nice = 0
 C process p_nice = 0
 D process p_nice = 5

 $p_cpu = p_cpu/2$
 $p_pri = P_USER + p_cpu/2 + 2*p_nice$

Starting point	A process		B process		C process		D process		Running processes	
	p_usrpri	p_cpu	p_usrpri	p_cpu	p_usrpri	p_cpu	p_usrpri	p_cpu	before	after
1	60	0	60	0	60	0	60	0		A
2	60	10	60	0	60	0	60	0	A	B
10	60	10	60	10	60	10	60	0	B	C
20	60	10	60	10	60	10	60	0	C	D
30	60	10	60	10	60	10	60	10	D	A
40	60	20	60	10	60	10	60	10	A	B
50	60	20	60	20	60	10	60	10	B	C
60	60	20	60	20	60	20	60	10	C	D
70	60	20	60	20	60	20	60	20	D	A
80	60	30	60	20	60	20	60	20	A	B
90	60	30	60	29	60	20	60	20	B	C
99	66	25	66	25	64	17	74	17	C	A
100	66	25	66	25	64	18	74	17		
101	66	25	66	25	64	27	74	17	A	B
110	66	35	66	25	64	27	74	17	B	D
120	66	35	66	35	64	27	74	27	D	C
130	66	35	66	35	64	37	74	27	C	A
140	66	45	66	35	64	37	74	27	A	B
150	66	45	66	45	64	37	74	27	B	D
160	66	45	66	45	64	37	74	37	D	C
170	66	47	66	47	64	47	74	37	C	A
180	66	55	66	45	64	47	74	37		
190	78	47	76	39	74	39	91	31	A	B
199	78	47	76	39	74	41	91	31		
200										
201										

Kf $2*3/(2*3+1)$ 0,85
 0,85

p_usrpri 66,25

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.

Mentés: neptunkod_tobbszignal.c



```
#include<stdio.h>
#include<signal.h>
#include <stdlib.h>

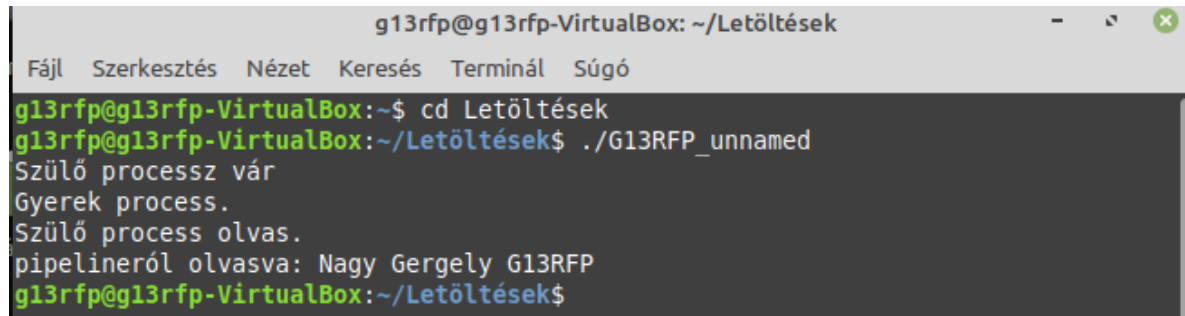
int sigint_counter = 0;
void handle_signals(int sig) {
    if (sig == 3) {
        printf("Quit signal caught.\n");
        exit(0);
    } else if (sig == 2) {
        if (sigint_counter == 0) {
            printf("First SIGINT.\n");
            sigint_counter++;
            signal(SIGINT, SIG_DFL);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

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

    while (1) ;
    return 0;
}
```

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.

Mentés: neptunkod_unnamed.c



```
g13rfp@g13rfp-VirtualBox: ~/Letöltések
Fájl Szerkesztés Nézet Keresés Terminál Súgó
g13rfp@g13rfp-VirtualBox:~$ cd Letöltések
g13rfp@g13rfp-VirtualBox:~/Letöltések$ ./G13RFP_unnamed
Szülő processz vár
Gyerek processz.
Szülő process olvas.
pipelineről olvasva: Nagy Gergely G13RFP
g13rfp@g13rfp-VirtualBox:~/Letöltések$
```

```
#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;
    }

    /*Szulo processz*/
    else if (p > 0) {
        char str[100];

        printf("Szülő processz vár\n");

        wait(NULL);

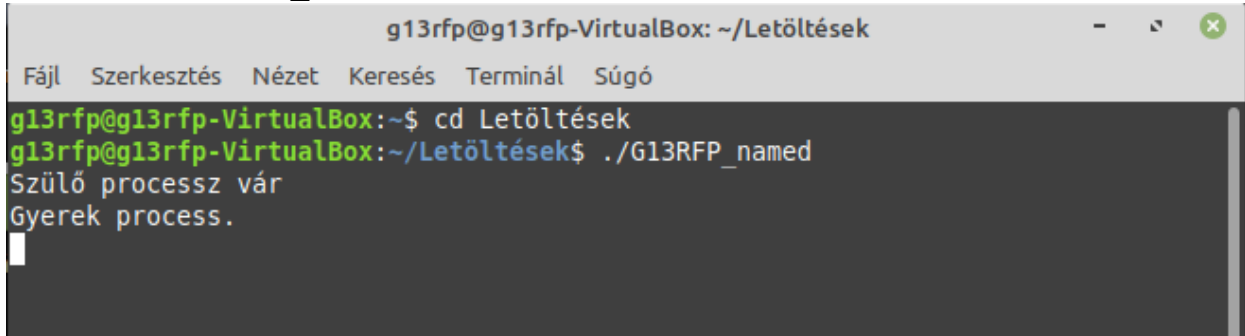
        printf("Szülő process olvas.\n");

        read(pipe1[0], str, 100);
        printf("pipelineről olvasva: %s\n", str);
        close(pipe1[0]);
    }
}
```

```
else {  
  
    printf("Gyerek process.\n");  
    char output_string[100];  
    strcpy(output_string, "Nagy Gergely G13RFP");  
    write(pipel[1], output_string, strlen(output_string) +  
1);  
  
    close(pipel[1]);  
  
    exit(0);  
}  
}
```

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.

Mentés: neptunkod_named.c



```
g13rfp@g13rfp-VirtualBox: ~/Letöltések
Fájl Szerkesztés Nézet Keresés Terminál Súgó
g13rfp@g13rfp-VirtualBox:~$ cd Letöltések
g13rfp@g13rfp-VirtualBox:~/Letöltések$ ./G13RFP_named
Szülő processz vár
Gyerek process.

```

```
#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* fifoname = "./G13RFP";
    mkfifo(fifoname, 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("Szülő processz vár\n");

        wait(NULL);

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

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

        printf("A %s nevű piperól olvasva: %s\n", fifoname,
```

```
str);  
  
    }  
    else {  
  
        printf("Gyerek process.\n");  
  
        char output_string[80];  
        strcpy(output_string, "Nagy Gergely G13RFP\n");  
  
        pipe = open(fifoname, O_WRONLY);  
        write(pipe, output_string, strlen(output_string));  
        close(pipe);  
  
        printf("Gyerek process vége.\n");  
  
        exit(0);  
    }  
}
```


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?

- Határozza meg a folyamatok által igényelt erőforrások mátrixát?
- Határozza meg pillanatnyilag szabad erőforrások számát?
- Igazolja az egyes processzek végrehajtásának lehetséges sorrendjét – számolással?”

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

erőforrás 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
szabad	5	3	2								

induló készlet: {5,3,2} ez P1 vagy P2 igényére elég

Válasszuk mondjuk P1-et,akkor:

$$\{5,3,2\} - \{2,2,2\} + \{2,2,2\} + \{2,2,3\} = \{5,3,2\} + \{2,2,3\} = \{7,5,5\}$$

új készlet: {7,5,5}

10 9 12

Foglalási igény				Maximális igény				Várható igény			
	R1	R2	R3		R1	R2	R3		R1	R2	R3
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
szabad	7	5	5								

válasszuk P2-t

$$\{7,5,5\} - \{0,2,1\} + \{0,2,1\} + \{1,2,2\} = \{7,5,5\} + \{1,2,2\} = \{8,7,7\}$$

új készlet: {8,7,7}

10 9 12

Foglalási igény				Maximális igény				Várható igény			
	R1	R2	R3		R1	R2	R3		R1	R2	R3
P3	0	1	3	P3	6	7	7	P3	6	6	4
P4	2	1	2	P4	3	7	10	P4	1	6	8
szabad	8	7	7								

10 9 12

Foglalási igény				Maximális igény				Várható igény			
	R1	R2	R3		R1	R2	R3		R1	R2	R3
P4	2	1	2	P4	3	7	10	P4	1	6	8
szabad	8	8	10								

10 9 12

Foglalási igény				Maximális igény				Várható igény			
	R1	R2	R3		R1	R2	R3		R1	R2	R3
	0	0	0		0	0	0		0	0	0
szabad	10	9	12								