Operációs rendszerek BSc

3. Gyak.

2022. 04. 27.

Készítette:

Kiss Bence Bsc

Mérnökinformatika

BYO2P7

Miskolc, 2022

1. feladat - 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

RR nélkül:

	A Process		B Process		C Process		D Process		Reschedule	
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	60	0	60	0	60	0		A
2	75	30	60	0 ... 60	60	0	60	0	A	B
3	67	15	75	30	60	0 ... 60	60	0	B	C
4	63	7	67	15	75	30	60	0 ... 60	C	D
5	61	3 ... 63	63	7	67	15	80	40	D	A
6	75	31	61	3 ... 63	63	7	70	20	D	B
7	67	15	75	31	61	3 ... 63	65	10	B	C
8	63	7	67	15	75	31	62	5 ... 65	C	D

RR-nal:

	A Process		B Process		C Process		D Process		Reschedule	
Clock Tick	p->spri	p->cpu	p->spri	p->cpu	p->spri	p->cpu	p->spri	p->cpu	running before	running after
Starting Point	60	0	60	0	60	0	60	0		A
1		1								
2		2								
...										
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
99	60	30	60	29	60	20	60	20		
100	66	25	66	**25**	64	17	74	17	B	C
101	66	25	66	25	64	18	74	17		
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			64	**47**	74	37	C	A
199	66	55	66	45	64	47	74	37		
200	78	**47**	76	39	74	39	91	31	A	B
201	78	47	76	39	74	41	91	31		

2. feladat - 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 sigintCount = 0;
```

```
void handleSignals(int sig) {
    if (sig == 3) {
        printf("SIGQUIT\n");
        exit(0);
    } else if (sig == 2) {
        if (sigintCount == 0) {
            printf("SIGINT\n");
            sigintCount++;
            signal(SIGINT, SIG_DFL);
        }
    }
}


int main()
{
    signal(SIGINT, handleSignals);
    signal(SIGQUIT, handleSignals);
    while (1) ;
    return 0;
}
```

3. feladat - 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. Mentés: neptunkod_unnamed.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;
    }


    else if (p > 0) {
        char str[100];
```

```
        printf("parent process vár\n");
        wait(NULL);
        printf("parent process olvas\n");
        read(pipe1[0], str, 100);
        printf("Pipeline-ról olvasva: %s\n", str);
        close(pipe1[0]);
    }
    else {

        printf("child process\n");
        char output_string[100];
        strcpy(output_string, "Kiss Bence");
        write(pipe1[1], output_string, strlen(output_string) + 1);
        close(pipe1[1]);
        exit(0);
    }
}
```

4. feladat - 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. Mentés: neptunkod_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* fifoname = "/tmp/BYO2P7";
    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("parent process vár\n");
        wait(NULL);
        printf("parent process olvas\n");
        pipe = open(fifoname, O_RDONLY);
        read(pipe, str, 80);
```

```
        close(pipe);
        printf("%s piperól olvasva: %s\n", fifoname, str);

    }
    else {

        printf("child process\n");


        char output_string[80];
        strcpy(output_string, "Kiss Bence\n");
        pipe = open(fifoname, O_WRONLY);
        write(pipe, output_string, strlen(output_string));
        close(pipe);
        printf("child process end\n");
        exit(0);
    }


    return 0;
}
```

5. feladat - 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?

2	2	2
0	2	1
6	6	4
1	6	8

b) Határozza meg pillanatnyilag szabad erőforrások számát?

Szabad erőforrások:	10	9	12	-	5	6	10	-	5	3	2

max_r	10	9	12