Jegyzőkönyv Operációs rendszerek

5. gyakorlat

1.feladat

Timkó András
HZS05V
ge-BGI

1. A `system()` rendszerhívással hajtson végre egy parancsot.

```
 GNU nano 5.4                                                hzs05v1fel.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
        char vegrehajto[20];

        strcpy( vegrehajto, "ls -l" );
        system(vegrehajto);

        return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# nano hzs05v1fel.c
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# gcc hzs05v1fel.c -o hzs05v1fel
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v1fel
total 24
-rwxr-xr-x 1 root root 16616 márc   7 17:44 hzs05v1fel
-rw-r--r-- 1 root root   167 márc   7 17:44 hzs05v1fel.c
```

2. Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre

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

int main () {
        char str[10];

        printf( "Enter a command\n" );
        fgets(str, 20, stdin);
        system(str);
        return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# nano hzs05v2fel.c
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# gcc hzs05v2fel.c -o hzs05v2fel
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v2fel
Enter a command
ls
hzs05v1fel  hzs05v1fel.c  hzs05v2fel  hzs05v2fel.c
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v2fel
Enter a command
date
2022. márc. 7., hétfő, 17:56:03 CAT
```

3. Készítsen egy XY_parent.c és a XY_child.c programokat. A XY_parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (10-ször)

```
 GNU nano 5.4                                          hzs05v_parent.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {

                system("./hzs05v_child");
                return(0);

}
```

```
 GNU nano 5.4                                          hzs05v_child.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
                int i;
                for(i=1; i<=10; i++){
                printf("hello\n");
                }
                return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v_parent
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
```

4. A `fork()` rendszerhívással hozzon létre egy gyerek processzt-t és abban hívjon meg egy `exec` családbeli rendszerhívást (pl. `execlp`).

```
  GNU nano 5.4                                                hzs05v4fel.c *
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {
                char *argv[3] = {"Command-line", ".", NULL};
                int pid = fork();
                if ( pid == 0 ) {
                execvp( "find", argv );

                }else{
                printf("Ez egy szulo process \n a gyereke: %d\n ", pid);
                wait(2);
        }


                return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v4fel
Ez egy szulo process
 a gyereke: 2933
.
./hzs05v1fel
./hzs05v_parent
./hzs05v_child
./hzs05v_parent.c
./hzs05v2fel.c
./hzs05v1fel.c
./hzs05v4fel.c
./hzs05v2fel
./hzs05v_child.c
./hzs05v4fel
 root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307#
```

5. A `fork()` rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekben: `exit, abort, nullával való osztás`)!

```
 GNU nano 5.4                                                hzs05v5fel.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {

                int pid = fork();
                if ( pid == 0 )
                 {
                        exit(999);

                 }

                int status;
                waitpid(pid, &status, 0);

                if ( WIFEXITED(status)) {

                int exit_status = WEXITSTATUS(status);
                printf("Exit kod: %d\n", exit_status);
                }
return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v5fel
Exit kod: 231
```

```
 GNU nano 5.4                                                hzs05v5_1fel.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main () {

                FILE *pf = fopen("nemletezik.txt","r");
                if (pf == NULL) {
fprintf(stderr, "hiba a file megnyitasaval\n");
abort();
}
return(0);

}
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v5_1fel
hiba a file megnyitasaval
Aborted
```

```
 GNU nano 5.4                                                hzs05v5_2fel.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
int main () {

                assert(1/0);
                return(0);

}
```

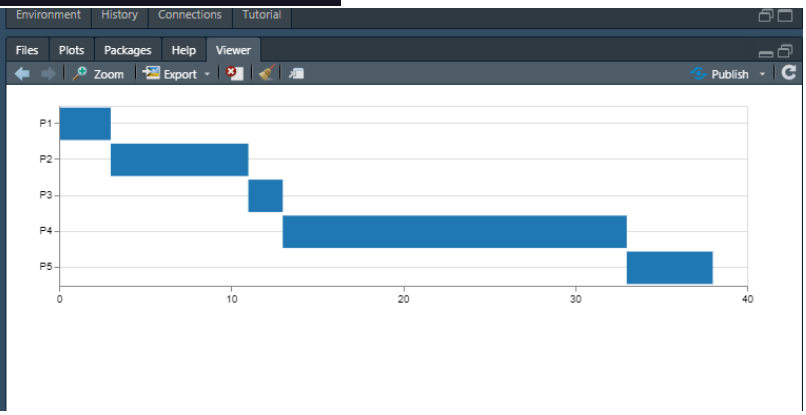
```
hzs05v5_2fel.c: In function ‘main’:
hzs05v5_2fel.c:7:11: warning: division by zero [-Wdiv-by-zero]
    7 |   assert(1/0);
      |           ^
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v5_2fel
Floating point exception
```

```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# nano hzs05v6.c
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# gcc hzs05v6.c -o hzs05v6
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v6
FCFS/Sorrend    CPU ido    Indulas/Varakozas    Befejezes
   1            3               0                   3
   2            8               3                   11
   3            2               11                   13
   4            20               13                   33
   5            5               33                   38
Atlag varakozasi ido = 12
Atlag befejezesi ido = 19
```

```
# https://candela.readthedocs.io/en/latest/index.html
library(candela)

data <- list(
  list(name='P1', level=1, start=0, end=3),
  list(name='P2', level=1, start=3, end=11),
  list(name='P3', level=1, start=11, end=13),
  list(name='P4', level=1, start=13, end=33),
  list(name='P5', level=1, start=33, end=38) ) ;


candela('GanttDiagramm',
        data=data, label='name',
        start='start', end='end', level='level',
        width=700, height=200)
```




```
// C program for FCFS
#include<stdio.h>
void findWaitingTime(int processes[], int n,
                          int bt[], int wt[])
{
    // varakozasi ido az elso processnek 0
    wt[0] = 0;

    // calculating  varakozasi ido
    for (int  i = 1; i < n ; i++ )
        wt[i] =  bt[i-1] + wt[i-1] ;
}

void findTurnAroundTime( int processes[], int n,
                  int bt[], int wt[], int tat[])
{
    // calculating befejzesi idp
    // bt[i] + wt[i]
    for (int  i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

//Function to calculate atlag ido
void findavgTime( int processes[], int n, int bt[])
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;


    findWaitingTime(processes, n, bt, wt);


    findTurnAroundTime(processes, n, bt, wt, tat);


    printf("FCFS/Sorrend   CPU ido   Indulas/Varakozas   Befejezes\n");



    for (int  i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf("   %d ",(i+1));
        printf("           %d ", bt[i] );
        printf("              %d",wt[i] );
        printf("                   %d\n",tat[i] );
    }
    int s=(float)total_wt / (float)n;
    int t=(float)total_tat / (float)n;
    printf("Atlag varakozasi ido = %d",s);
    printf("\n");
    printf("Atlag befejezesi ido = %d\n ",t);
}


int main()
{
    //process id's
    int processes[] = { 1, 2, 3, 4, 5};
    int n = sizeof processes / sizeof processes[0];

    //cpu idok
    int  burst_time[] = {3, 8, 2, 20, 5};

    findavgTime(processes, n,  burst_time);
    return 0;
}
```

I. Határozza meg FCFS és SJF esetén

a.) A befejezési időt?

b.) A várakozási/átlagos várakozási időt?

c.) Ábrázolja Gantt diagram segítségével *az aktív/várakozó processzek* futásának menetét.

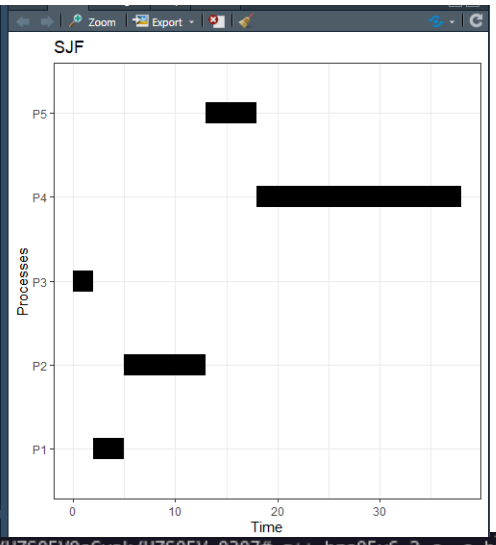
Megj.: a Gantt diagram ábrázolása szerkesztő program segítségével vagy Excel programmal.

FCFS	Érkezés	CPU idő
P1	0	3
P2	1	8
P3	3	2
P4	9	20
P5	12	5

```r
library(ggplot2)

data <- data.frame(name = c('P1', 'P2', 'P3', 'P4', 'P5'),
                   start = c(2, 5, 0, 18, 13),
                   end = c(5, 13, 2, 38, 18)
)

ggplot(data, aes(x=start, xend=end, y=name, yend=name)) +
  theme_bw()+
  geom_segment(size=8) +
  labs(title='SJF', x='Time', y='Processes')
```




```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# g++ hzs05v6_2.c -o hzs05v6_2
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v6_2
SJF         CPU ido       Varakozasi ido      Befejezsi ido
 1              3               0               3
 2              8               4               12
 3              2               0               2
 4              20              9               29
 5              5               1               6

Atlag varakozasi ido = 2.8
Atlag befejezesi ido = 10.4root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# nano hzs05v
```

```cpp
// C++ program to SJF
#include <bits/stdc++.h>
using namespace std;

struct Process {
    int pid;
    int bt;     //cpu ido
    int art; //erkezes
};


void findWaitingTime(Process proc[], int n,
                             int wt[])
{
    int rt[n];


    for (int i = 0; i < n; i++)
        rt[i] = proc[i].bt;

    int complete = 0, t = 0, minm = INT_MAX;
    int shortest = 0, finish_time;
    bool check = false;


    while (complete != n) {

        for (int j = 0; j < n; j++) {
            if ((proc[j].art <= t) &&
            (rt[j] < minm) && rt[j] > 0) {
                minm = rt[j];
                shortest = j;
                check = true;
            }
        }

        if (check == false) {
            t++;
            continue;
        }


        rt[shortest]--;


        minm = rt[shortest];
        if (minm == 0)
            minm = INT_MAX;


        if (rt[shortest] == 0) {


            complete++;
            check = false;

            finish_time = t + 1;
            wt[shortest] = finish_time -
                        proc[shortest].bt -
                        proc[shortest].art;

            if (wt[shortest] < 0)
                wt[shortest] = 0;
        }

        t++;
    }
}


void findTurnAroundTime(Process proc[], int n,
                                int wt[], int tat[])
{
    for (int i = 0; i < n; i++)
        tat[i] = proc[i].bt + wt[i];
}


void findavgTime(Process proc[], int n)
{
    int wt[n], tat[n], total_wt = 0,
                        total_tat = 0;

    findWaitingTime(proc, n, wt);

    findTurnAroundTime(proc, n, wt, tat);

    cout << "SJF         "
        << " CPU ido      "
        << " Varakozasi ido     "
        << " Befejezsi ido      "
        << " Erkezesi ido\n";

    for (int i = 0; i < n; i++) {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << proc[i].pid << "\t\t"
            << proc[i].bt << "\t\t " << wt[i]
            << "\t\t " << tat[i] << "\t\t" << proc[i].art << endl;
    }

    cout << "\nAtlag varakozasi ido = "
        << (float)total_wt / (float)n;
    cout << "\nAtlag befejezesi ido = "
        << (float)total_tat / (float)n;
}


int main()
{
    Process proc[] = { { 1, 3, 0 }, { 2, 8, 1 },
                       { 3, 2, 3 }, { 4, 20, 9 }, {5, 5, 12} };
    int n = sizeof(proc) / sizeof(proc[0]);

    findavgTime(proc, n);
    return 0;
}
```

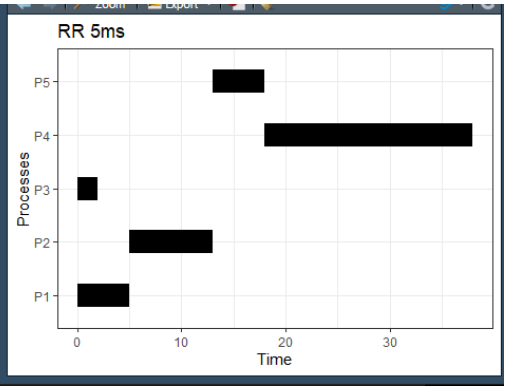
```
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# g++ hzs05v6_3.c -o hzs05v6_3
root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307# ./hzs05v6_3
Processzek  cpu ido  varakozas      Befejezesi ido
 1             3        0              3
 2             8        15             23
 3             2        8              10
 4             20       18             38
 5             5        15             20
Atlagos varakozasi ido  = 11.2
Atlagos befejezesi ido = 18.8root@elegemvan:/home/elegemvan/Desktop/HZS05VOsGyak/HZS05V_0307#
```

f

```
library(ggplot2)

data <- data.frame(name = c('P1', 'P2', 'P3', 'P4', 'P5'),
                   start = c(0, 5, 0, 18, 13),
                   end = c(5, 13, 2, 38, 18)
)

ggplot(data, aes(x=start, xend=end, y=name, yend=name)) +
  theme_bw()+
  geom_segment(size=8) +
  labs(title='RR 5ms', x='Time', y='Processes')
```




```
// C++ program for RR
#include<iostream>
using namespace std;

void findWaitingTime(int processes[], int n,
          int bt[], int wt[], int quantum)
{
    int rem_bt[n];
    for (int i = 0 ; i < n ; i++)
        rem_bt[i] = bt[i];

    int t = 0;

    while (1)
    {
        bool done = true;


        for (int i = 0 ; i < n; i++)
        {
            if (rem_bt[i] > 0)
            {
                done = false;

                if (rem_bt[i] > quantum)
                {
                    t += quantum;

                    rem_bt[i] -= quantum;
                }

                else
                {
                    t = t + rem_bt[i];

                    wt[i] = t - bt[i];

                    rem_bt[i] = 0;
                }
            }
        }

        if (done == true)
        break;
    }
}

void findTurnAroundTime(int processes[], int n,
                        int bt[], int wt[], int tat[])
{
    for (int i = 0; i < n ; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime(int processes[], int n, int bt[],
                                 int quantum)
{
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    findWaitingTime(processes, n, bt, wt, quantum);

    findTurnAroundTime(processes, n, bt, wt, tat);

    cout << "Processzek "<< " cpu ido "
        << " varakozas     " << " Befejezesi ido\n";

    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
            << wt[i] <<"\t\t " << tat[i] <<endl;
    }

    cout << "Atlagos varakozasi ido  = "
        << (float)total_wt / (float)n;
    cout << "\nAtlagos befejezesi ido = "
        << (float)total_tat / (float)n;
}


int main()
{

    int processes[] = { 1, 2, 3, 4, 5};
    int n = sizeof processes / sizeof processes[0];


    int burst_time[] = {3, 8, 2, 20, 5};


    int quantum = 5;
    findavgTime(processes, n, burst_time, quantum);
    return 0;
```

II. Round Robin (RR) esetén

a.) Ütemezze az adott időszelet (5ms) alapján az egyes processzek (befejezési és várakozási/átlagos várakozási idő) paramétereit (ms)!

b.) A rendszerben lévő processzek végrehajtásának sorrendjét?

c.) Ábrázolja Gantt diagram segítségével az *aktív/várakozó processzek* futásának menetét!”

RR: 5ms	Érkezés	CPU idő
P1	0	3
P2	1	8
P3	3	2
P4	9	20
P5	12	5