**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

по дисциплине «Операционные системы»

на тему: **«Программирование планировщиков процессов»**

Выполнил:студент гр. ИТИ-11

Щербенко М.А.

Принял: преподаватель-стажёр

Карась О. В.

Гомель 2022

**Цель работы:** разработать программу, осуществляющую моделирование режима работы с разделением времени.

**Задание.** В соответствии с вариантом (выдается преподавателем), разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов.

|  |  |
| --- | --- |
| **Вариант 11** | Алгоритм SJF. Не вытесняющий, приоритетный |

Результат работы программы:

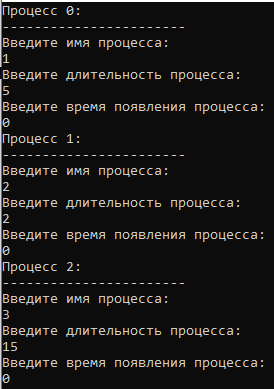


Рисунок 1 – “Выполнение программы на языке с”

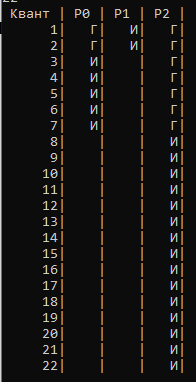


Рисунок 2 – “Выполнение программы на языке с”

**Вывод**: в ходе выполнения лабораторной работы были изучены такие способ синхронизации процессов как: переменная-замок, строгое чередование, алгоритм булочной (Bakery algorithm).

**Листинг программы:**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

char name1[10],name2[10],name3[10];

int duration1, duration2, duration3;

int prior1,prior2,prior3;

int time1, time2, time3;

struct processes

{

char name[10];

int duration;

int prior;

int time;

char symb[5];

int left;

};

int main()

{

int minPriority,minDuration,currentInd;

setlocale(LC\_ALL,"RUS");

struct processes process[3];

for (int i=0;i<3;i++)

{

printf("Ïðîöåññ %d: \n-----------------------\n",i);

fflush(stdin);

puts("Ââåäèòå èìÿ ïðîöåññà: ");

gets(process[i].name);

puts("Ââåäèòå äëèòåëüíîñòü ïðîöåññà: ");

scanf("%d",&process[i].duration);

puts("Ââåäèòå âðåìÿ ïîÿâëåíèÿ ïðîöåññà: ");

scanf("%d",&process[i].time);

process[i].left=process[i].duration;

process[i].prior=0;

}

int timeAll=process[0].duration+process[1].duration+process[2].duration;

printf("%d\n",timeAll);

puts(" Êâàíò | P0 | P1 | P2 |");

for (int i=1;i<=timeAll;i++)

{

minPriority=100;

minDuration=timeAll;

for(int k=0;k<3;k++)

{

if (process[k].time<=i && process[k].left>0)

{

if (process[k].prior<=minPriority)

{

minPriority=process[k].prior;

if (process[k].duration<minDuration)

{

minDuration=process[k].duration;

currentInd=k;

}

}

}

}

for (int k=0;k<3;k++)

{

if(currentInd==k && process[k].left>0)

strcpy(process[k].symb,"È");

else if(process[k].time<=i && process[k].left>0)

strcpy(process[k].symb,"Ã");

else

strcpy(process[k].symb," ");

}

//printf("%7d|%4s|%4s|%4s|\n",i,process[0].symb,process[1].symb,process[2].symb);

printf("%7d|%4s|%4s|%4s|\n",i,process[0].symb,process[1].symb,process[2].symb);

process[currentInd].left--;

}

fflush(stdin);

getchar();

return 0;

}