Учреждение образования Республики Беларусь

«Гомельский государственный технический

университет им. П.О.Сухого»

Кафедра «Автоматизированных и информационных систем»

Лабораторная работа № 6

ПРОГРАММИРОВАНИЕ ПЛАНИРОВЩИКОВ ПРОЦЕССОВ

Выполнил: студент группы ИТИ-11

Тесёлкин Б.А.

Принял: преподаватель

Соболев Д.В.

Гомель 2018

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

**Задание.**

Разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов.

На экран выводится следующая информация:

1. Номер текущего кванта времени процессора;

2. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления;

3. Таблица планирования процессов с отображением текущего состояния процессов.

После запуска, программа должна диалоговом режиме, ввести информацию процессах – имя, длительность, приоритет, время появления. Для алгоритмов RR число квантов времени. Выполнение должно производиться в пошаговом режиме (по нажатию на кнопку). По окончанию работы процесса на экране должно выводится сообщение о его завершении.

Алгоритм: **Round Robin ( приоритетный )**

Язык: С

Листинг программы

#include <stdio.h>

#include <conio.h>

int main()

{

int i,m,quanta,length=0,j,count=0,count3=0,nameNum = -1,first,min,k;

bool flag = 1,flag3;

puts("Enter the number of processes: ");

scanf("%d", &m);

int \*duration = new int[m];

int \*turn = new int[m];

int \*appear = new int[m];

int \*priorities = new int[m];

char \*\*a = new char \*[m];

for(i = 0; i < m; i++)

a[i] = new char[100];

puts("Enter processes names: ");

gets(a[0]);

for(i = 0; i < m; i++)

gets(a[i]);

puts("Enter the burst time: ");

for(i = 0; i < m; i++)

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

puts("Enter processes priorities: ");

for(i = 0; i < m; i++)

scanf("%d",&priorities[i]);

puts("Enter the tact:");

scanf("%d",&quanta);

printf("\xDA\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC2");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC2");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xBF\n");

printf("\xB3 N \xB3");

for(i=1;i<=m;i++)

{

printf(" %2.d \xB3",i);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

printf("\xB3 Name \xB3");

for(i=1;i<=m;i++)

{

printf(" %s \xB3",a[i-1]);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

printf("\xB3 Priorities \xB3");

for(i=0;i<m;i++)

{

printf(" %2.d \xB3",priorities[i]);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

printf("\xB3 Burst Time \xB3");

for(i=0;i<m;i++)

{

printf(" %2.d \xB3",duration[i]);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC1");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

printf("\xB3 Quantum \xB3");

printf(" %2.d \xB3\n",quanta);

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC4");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xD9\n");

printf("\xDA\xC4\xC4\xC4\xC4\xC4\xC4\xC2");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC2");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xBF\n");

printf("\xB3 N \xB3");

for(i=1;i<=m;i++)

{

printf(" %2.d \xB3",i);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

printf("\xB3 Name \xB3");

for(i=1;i<=m;i++)

{

printf(" %s \xB3",a[i-1]);

}

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<m;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

for(i=0;i<m;i++)

for(j=i+1;j<m;j++)

{

if(priorities[i] == priorities[j]) priorities[j]++;

}

for(i = 0; i < m; i++)

length = length + duration[i];

for(k = 0; k < m;k++) turn[k] = 1;

for(i = 0;i<=length;i++)

{

if(i == 0) printf("\xB3 0 \xB3");

else printf("\xB3 %2.0d \xB3",i);

for(j = 0;j < m;j++)

{

min = 100;

for(k = 0; k < m;k++) if (priorities[k] == 0) turn[k] = 0;

for(k = 0; k < m;k++)

if(turn[k] == 0)

{

flag3 = 1;

}

else

{

flag3 = 0;

break;

}

if( flag3 == 1 || count3 ==m-1 )

{

for(k = 0; k < m;k++) turn[k] = 1;

flag3 = 0;

}

for(k = 0; k < m;k++) if(priorities[k] <= min && priorities[k] > 0 && turn[k] != 0) min = priorities[k];

for(k = 0; k < m;k++) if(min == priorities[k]) first = k;

if(duration[j] > 0 && flag == 1 && j == first)

{

printf(" W \xB3");

duration[j] = duration[j] - 1;

flag = 0;

count++;

if(count == quanta && duration[j] > 0)

{

turn[j] = 0;

count = 0;

}

else

if(count <= quanta && duration[j] == 0)

{

count3++;

turn[j] = 0;

priorities[j] = 0;

count = 0;

nameNum = j;

}

}

else

if(duration[j] > 0) printf(" R \xB3");

else

printf(" \xB3");

}

if( nameNum!=-1 ) printf("The process %s is over", a[nameNum]);

printf("\n");

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(j=1;j<m;j++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xC4\xB4\n");

nameNum = -1;

flag = 1;

getch();

}

return(0);

printf(“The end”);

getch();

}

Запустим программу и введём исходные данные.

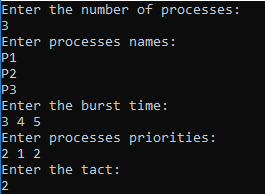


Рисунок 1 – Запрос ввода данных

После запуска программы на экран будет выведена информация о процессах после истечения первого кванта времени, кол-во оставшихся квантов времени для выполнения текущего процесса, а так же таблица планирования процессов с отображением текущего состояния процессов (W – исполнение, R - готовность).

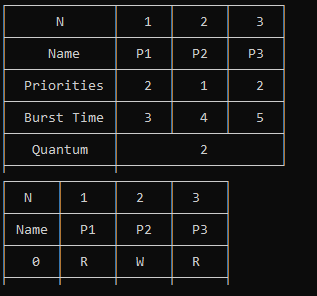


Рисунок 2 – Информация о ходе планирования

По нажатию на клавишу “Enter” информация будет обновляться. Вскоре, когда один из процессов будет завершён, на экран выведется соответствующая информация. В таблице это будут иллюстрировать пустые заполнения.

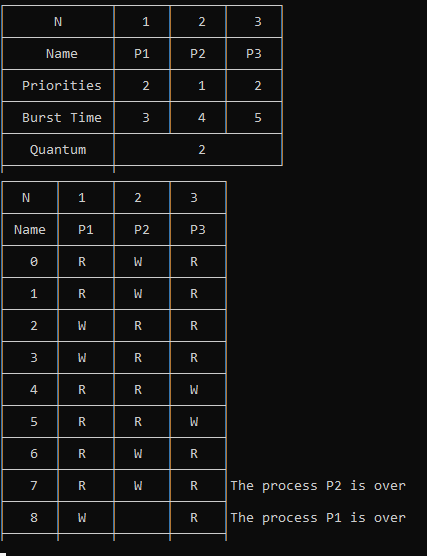


Рисунок 3 – Отражение завершения процесса на таблице

После завершения всех процессов программа прекратит своё выполнение и на экран выведется соответствующая информация.

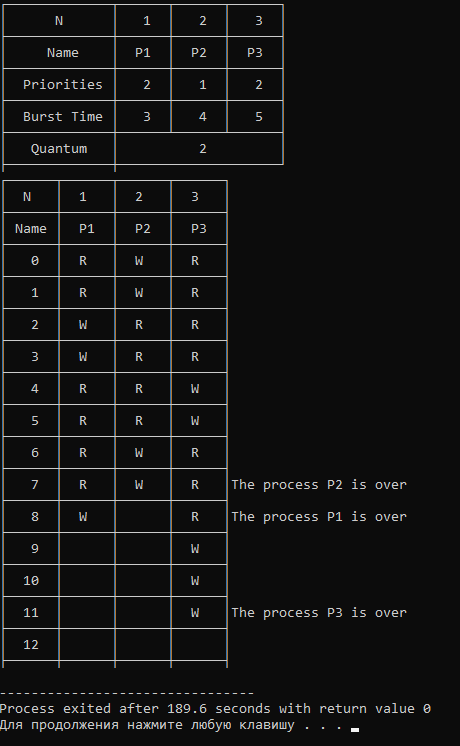


Рисунок 5 – Конечная информация по завершению программы

**Вывод:** в процессе выполнения данной лабораторной работы разработали программу, осуществляющую моделирование режима работы с разделение времени.