Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Операционные Системы Реального Времени

Лабораторная работа №2

Процессы и асинхронное взаимодействие

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: профессор кафедры ПМиК

Фионов Андрей Николаевич

Новосибирск, 2020

**Содержание**

1. Задание.
2. Описание работы
3. Текст программы
4. Результаты работы

**Задание**

1. Тщательно изучить библиотеку VinGraph.
2. Используя функции библиотеки VinGraph, нарисовать абстрактную картину, которой представлены (почти) все доступные графические элементы.
3. Заставить нарисованные элементы двигаться независимо друг от друга с помощью параллельных процессов (можно изменять во времени положение, цвет, размеры, конфигурацию графических элементов). Предусмотреть завершение программы по нажатию на любую клавишу.
4. Нарисовать нечто, движущееся по замкнутой кривой. Организовать изменение траектории движения по нажатию на клавиши (организуя взаимодействие процессов через общую область памяти (shared memory)). В качестве фона можно использовать (оживленную) картину, созданную на предыдущих этапах работы.
5. Затем последнюю программу сделать с помощью нитей в одном процессе.

**Описание работы**

Лабораторная работа представлена в виде двух программ – fork.cpp и pthread.cpp. В них реализованы движение элементов с помощью процессов и потоков соответственно. В качаестве абстрактной картины используются 10 квадратов(rectangle), двигающихся с разной скоростью и имеющих разный цвет, а также эллипс, двигающийся по замкнутой кривой (по окружности).

Движение квадратов реализовано в функции rectangle\_move, эллипса – thing\_move. В программе fork.cpp для взаимодейтсвия с эллипсом по нажатию стрелок используется разделяемая память (mmap).

Параметры объектов а также их идентификатор процесса (потока) описаны в структуре object.

Текст программы

fork.cpp

#include <math.h>

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include <vingraph.h>

#include <sys/mman.h>

Struct object

{

Int pid;//ид процесса(потока)

Int type;//тип объекта

Int dx;//движение по оси x(начальная координата для эллипса)

Int dy;//движение по оси y(начальная координата для эллипса)

Int color;//цвет

Float phi;//не используется для rectangle

}\*ptr;

Void rectangles\_move (int obj, int dx, int dy)

{

tPoint current\_position;//текущее положение объекта

tPoint screen\_dimension = GetDim(0);//координаты экрана

tPoint object\_dimension = GetDim(obj);//координаты объекта

for(;;)

{

Move (obj, dx, dy);

Current\_position = GetPos(obj);//получаем позицию объекта

If (current\_position.x <= 0 || current\_position.x >= (screen.dimension.x – object\_dimension.x))//если объект уходит за экран

{

Dx \*= -1;

}

If (current\_position.y <= 0 || current\_position.y >= (screen.dimension.y – object\_dimension.y))//если объект уходит за экран

{

Dy \*= -1;

}

Usleep (1000);

}

}

Void thing\_move (int obj, int a, int b)

{

Int x = 0;

Int y = 0;

Float rho = 0.0;

Float phi = 0.0;

For (;;)

{

Rho = 5 \* cos (phi) + b;

X = rho \* cos (phi) + a;

Y = rho \* sin (phi) + b;

MoveTo (x, y, obj);

Phi += ptr->phi;

Usleep(1000);

}

}

Int main(int argc, char \*argv[])

{

object objects[10];

int colors[10] =

{

RGB(13, 123, 16),

RGB(133, 15, 85),

RGB(5, 230, 210),

RGB(45, 112, 32),

RGB(234, 156, 43),

RGB(32, 178, 122),

RGB(123, 132, 32),

RGB(180, 199, 121),

RGB(40, 44, 233),

RGB(87, 15, 150)

}

ConnectGraph();

Fill (0, colors[0]); //заполнение фона

For (int I = 0; I < 10; i++)

{

objects[i].pid = fork();

If (objects[i].pid == 0)

{

objects[i].type = 0;

objects[i].dx = I;

objects[i].dy = I;

objects[i].color = colors[i];

objects[i].type = Rect(150, 150, 50, 50);

objects[i].phi = 0.0;

Fill (objects[i].type, objects[i].color);

Rectangles\_move(objects[i].type, objects[i].dx, objects[i].dy);

}

}

Ptr = (object\*)mmap(0, sizeof(object), PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANON, -1, 0);

If (ptr == MAP\_FAILED)

{

Return -1;

}

Ptr->pid = fork();

If (ptr->pid == 0)

{

Ptr->dx = 100;

Ptr->dy = 200;

Ptr->type = Ellipse(ptr->dx, ptr->dy, 50, 50);

Ptr->color = RGB(255, 255, 255);

Ptr->phi = 0.01;

Thing\_move(ptr->type, ptr->dx, ptr->dy);

}

Int ch;

While (ch != 27)

{

Ch = InputChar();

Switch(ch)

{

Case 81:

Ptr->phi = abs(ptr->phi) \* (-1);

Break;

Case 82:

If(ptr->phi)

{

Ptr->phi += 0.01;

}

Else

{

Ptr->phi -= 0.01;

}

Break;

Case 83:

Ptr->phi = abs(ptr->phi);

Break;

Case 84:

If(ptr->phi)

{

Ptr->phi -= 0.01;

}

Else

{

Ptr->phi += 0.01;

}

Break;

default:

Break;

}

}

For(int I = 0; I < 10; i++)

{

Kill(objects[i].pid, SIGTERM);

}

Kill(ptr->pid, SIGTERM);

CloseGraph();

Return 0;

}

Pthread.cpp

#include <math.h>

#include <stdio.h>

#include <signal.h>

#include <unistd.h>

#include <vingraph.h>

#include <sys/mman.h>

Struct object

{

Int pid;

Int type;

Int dx;

Int dy;

Int color;

Float phi;

};

Void \*rectangles\_move (object \*p)

{

Int dx = p->dx;

Int dy = p->dy;

Int obj = p->type;

tPoint current\_position;

tPoint screen\_dimension = GetDim(0);

tPoint object\_dimension = GetDim(obj);

for(;;)

{

Move (obj, dx, dy);

Current\_position = GetPos(obj);

If (current\_position.x <= 0 || current\_position.x >= (screen.dimension.x – object\_dimension.x))

{

Dx \*= -1;

}

If (current\_position.y <= 0 || current\_position.y >= (screen.dimension.y – object\_dimension.y))

{

Dy \*= -1;

}

Usleep (1000);

}

}

Void \*thing\_move (object \*p)

{

Int a = p->dx;

Int b = p->dy;

Int obj = p->type;

Int x = 0;

Int y = 0;

Float rho = 0.0;

Float phi = 0.0;

For (;;)

{

Rho = 5 \* cos (phi) + b;

X = rho \* cos (phi) + a;

Y = rho \* sin (phi) + b;

MoveTo (x, y, obj);

Phi += ptr->phi;

Usleep(1000);

}

}

Int main(int argc, char \*argv[])

{

object objects[10];

int colors[10] =

{

RGB(13, 123, 16),

RGB(133, 15, 85),

RGB(5, 230, 210),

RGB(45, 112, 32),

RGB(234, 156, 43),

RGB(32, 178, 122),

RGB(123, 132, 32),

RGB(180, 199, 121),

RGB(40, 44, 233),

RGB(87, 15, 150)

}

ConnectGraph();

Fill (0, colors[0]); //заполнение фона

For (int I = 0; I < 10; i++)

{

objects[i].pid = fork();

If (objects[i].pid == 0)

{

objects[i].type = 0;

objects[i].dx = I;

objects[i].dy = I;

objects[i].color = colors[i];

objects[i].type = Rect(150, 150, 50, 50);

objects[i].phi = 0.0;

Fill (objects[i].type, objects[i].color);

Pthread\_create(&objects[i].thread, 0, (void\*(\*)(void\*))rectangles\_move, &objects[i]);

}

}

Object thing;

Thing.thread = 0;

Thing.dx = 100;

Thing.dy = 200;

Thing.color = RGB(255,255,255);

Thing.type = Ellipse(thing.dx, thing.dy, 50, 50);

Thing.phi = 0.01;

Pthread\_create(&thing.thread, 0, (void\*(\*)(void\*))thing\_move, &thing);

Int ch;

While (ch != 27)

{

Ch = InputChar();

Switch(ch)

{

Case 81:

Thing.phi = abs(Thing.phi) \* (-1);

Break;

Case 82:

If(Thing.phi)

{

Thing.phi += 0.01;

}

Else

{

Thing.phi -= 0.01;

}

Break;

Case 83:

Thing.phi = abs(Thing.phi);

Break;

Case 84:

If(Thing.phi)

{

Thing.phi -= 0.01;

}

Else

{

Thing.phi += 0.01;

}

Break;

default:

Break;

}

}

For(int I = 0; I < 10; i++)

{

Pthread\_cancel(objects[i].thread);

}

Pthread\_cancel(thing.thread);

CloseGraph();

Return 0;

}

**Результаты работы**

