Лекция11

Обмен данными между процессами

1. Разделы dll с разделяемым доступом:
   а) стандартные и пользовательские разделы, реализация MS Visual Studio;
2. Синхронизация потоков различных процессов.
   а) барьерная синхронизация, WaitForMultipleObjects;
   б) синхронизация взаимоисключения, event, mutex, semaphore, реализация MS Windows;

Разделы *dll* с разделяемым доступом

libd1.cpp

```
#include <windows.h>
#pragma data_seg(".M_SH")
int d[10] = { 0 }; //инициализация обязательна
double s = 0.0;
#pragma data_seg()
```

libd1.def

```
LIBRARY      libd1
EXPORTS
        d     @1
        s     @2
```

```
> cl  /c  libd1.cpp

> link  /DLL /DEF:libd1.def /SECTION:.M_SH,RWS  libd1.obj
```

libd1.cpp

```
#include <windows.h>
#pragma data_seg(".M_SH")
int d[10] = { 0 }; //инициализация обязательна
double s = 0.0;
#pragma data_seg()
#pragma comment(linker, "/SECTION:.M_SH,RWS" )
```

```
> cl  /c  libd1.cpp

> link  /DLL /DEF:libd1.def  libd1.obj
```

pd1.cpp

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "libd1" )
extern __declspec(dllimport)
int d[10];
extern __declspec(dllimport)
double s;
int main() {
	int i;
	for (i = 0; i<10; i++)
		d[i] = i*i;
	s = 3.1415;

	getchar();
	return 0;
}
```

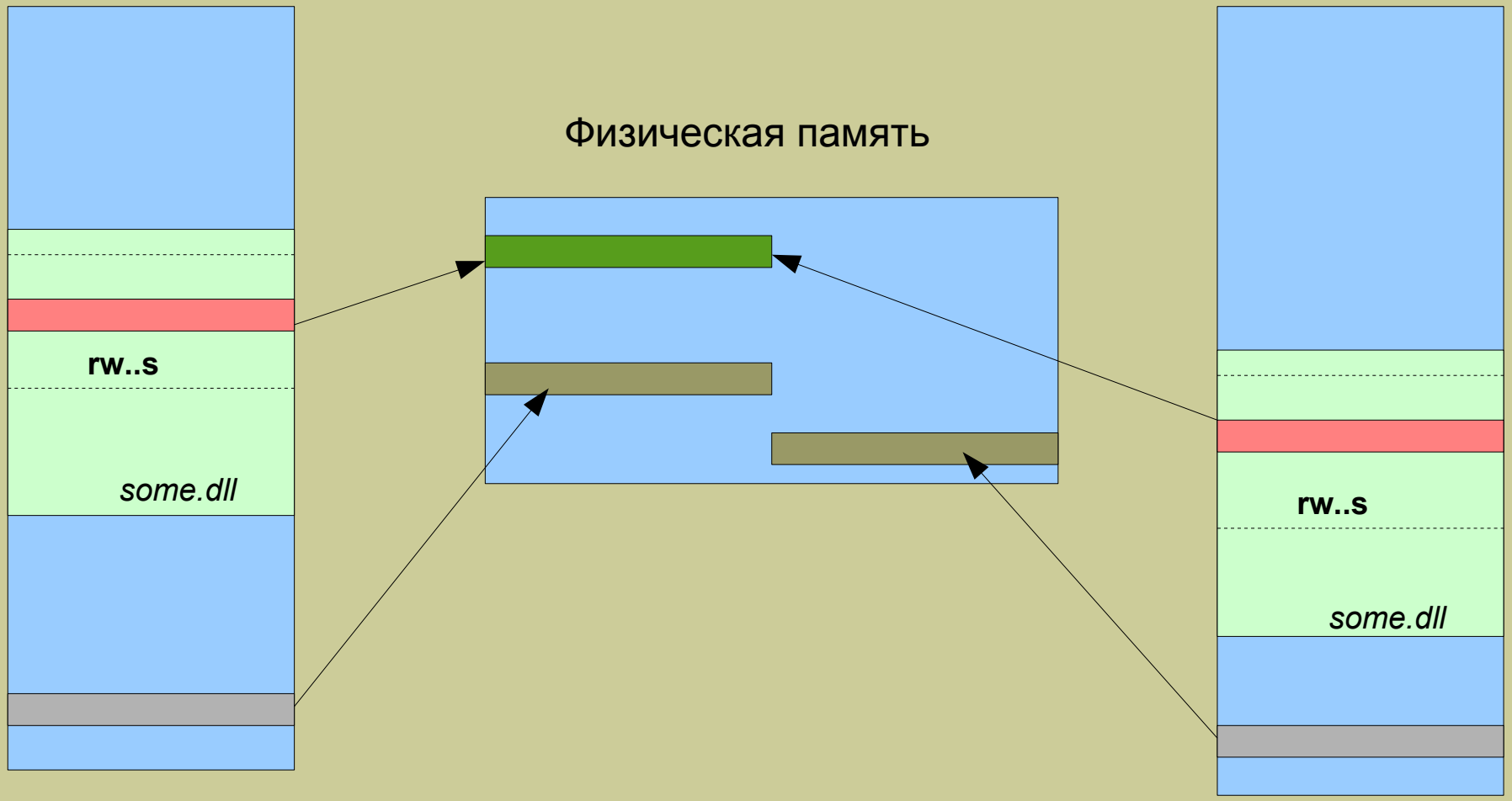
```
> cl pd1.cpp

> cl pd11.cpp
```

pd11.cpp

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "libd1" )
extern __declspec(dllimport)
int d[10];
extern __declspec(dllimport)
double s;
int main() {
	int i;
	for (i = 0; i < 10; i++)
		printf("%d\n", d[i]);
	printf("%g\n",s);
	return 0;
}
```

Адресное пространство процесса 1

Адресное пространство процесса 2



Структуры *C* с разделяемым доступом

libd2.cpp

```
#include <windows.h>

#pragma data_seg(".M_SH")
//extern __declspec(dllexport)
struct _shareTest {
	double g;
	int n;
}shareTest = {0.0, 1};
#pragma data_seg()
#pragma comment(linker, "/SECTION:.M_SH,RWS" )
```

libd2.def

```
LIBRARY	libd2
EXPORTS
	shareTest	@1
```

pd2.cpp

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "libd2" )

extern __declspec(dllimport)
struct _shareTest {
	double g;
	int n;
}shareTest;

int main() {
	shareTest.g = 2.78;
	shareTest.n = 127;

	getchar();
	return 0;
}
```

pd22.cpp

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "libd2" )

extern __declspec(dllimport)
struct _shareTest {
	double g;
	int n;
}shareTest;

int main() {
printf("%g\t%d\n", shareTest.g, shareTest.n);
return 0;
}
```

Барьерная синхронизация

```
#include <windows.h>
#include <stdio.h>

void main( void )
{
 STARTUPINFO si[3];
 PROCESS_INFORMATION pi[3];

 char* exefile[]={
            "C:\\Windows\\System32\\calc.exe",
            "C:\\Windows\\write.exe"
            };
 char* cmdLine="cmd /C C:\\Windows\\notepad.exe 1.cpp";

HANDLE Handles[3];

  for(int i=0;i<3;i++){
    ZeroMemory( &si[i], sizeof(si[i]) );
    si[i].cb = sizeof(si[i]);
    ZeroMemory( &pi[i], sizeof(pi[i]) );
  }
```

```
for(int i=0;i<2;i++){
   if( !CreateProcess(
       exefile[i], //"C:\\Windows\\notepad.exe", //Исполняемый модуль
       NULL,  // Командная строка не используется
       NULL,        // Дескриптор процесса не наследуется
       NULL,       // Дескриптор потока не наследуется
       FALSE,
      // Открытые дескрипторы родительского процесса не //наследуется
       CREATE_NO_WINDOW,
      // Не создавать окна (консольного окна)
       NULL,
      //  Используются переменные окружения родителя
       NULL,   // Используется текущая директория родителя
       &si[i],      // Указатель на структуру STARTUPINFO
       &pi[i] )
      // Указатель на структуру PROCESS_INFORMATION
    ){ printf("System error code: %i\n",GetLastError()); }

    Handles[i]=pi[i].hProcess;
 }
```

```
  if( !CreateProcess(
      NULL,
      cmdLine,  // Используется командная строка
      NULL,
      NULL,
      FALSE,
      CREATE_NO_WINDOW,
      NULL,
      NULL,
      &si[2],
      &pi[2] )
  ){ printf("System error code: %i\n",GetLastError()); }

  Handles[2]=pi[2].hProcess;

  WaitForMultipleObjects(3,Handles,TRUE, INFINITE);
  printf("Go home!");

  for(int i=0;i<3;i++){
    CloseHandle( pi[i].hProcess );
    CloseHandle( pi[i].hThread );
  }
}
```

Синхронизация потоков различных процессов.

libevd1.cpp

```
#include <windows.h>

#pragma data_seg(".M_SH")
extern __declspec(dllexport)
char sh[6] = { '\0' }; //инициализация обязательна
#pragma data_seg()
#pragma comment(linker, "/SECTION:.M_SH,RWS" )
```

ev2.c

```
#include <windows.h>
#include <process.h>
#include <stdio.h>
HANDLE hEvent1,hEvent2;
char sh[6];
void Thread( void* p);
int main( void ){
hEvent1=CreateEvent(NULL,FALSE,TRUE,NULL);
hEvent2=CreateEvent(NULL,FALSE,FALSE,NULL);
_beginthread( Thread, 0, NULL );
while( 1 ){
   WaitForSingleObject(hEvent1, INFINITE);
   printf("%s\n",sh);
   SetEvent(hEvent2);
}
CloseHandle(hEvent1);
CloseHandle(hEvent2);
return 0;
}
```

evd1.cpp

```
#include <windows.h>
//#include <process.h>
#include <stdio.h>
#pragma comment(lib, "libevd1" )
HANDLE hEvent1, hEvent2;
extern __declspec(dllimport)
char sh[6];
int main(void) {
   hEvent1 = CreateEvent(NULL, FALSE,
                         TRUE, "MyTestEvent1");
   hEvent2 = CreateEvent(NULL, FALSE,
                         FALSE, "MyTestEvent2");
    while (1) {
       WaitForSingleObject(hEvent1,
                                   INFINITE);
       printf("sh: %s\n", sh);
       SetEvent(hEvent2);
    }
    CloseHandle(hEvent1);
    CloseHandle(hEvent2);
    return 0;
}
```

```
void Thread( void* pParams ) {
  int counter = 0;
  while ( 1 ){
    WaitForSingleObject(hEvent2, IN
    if(counter%2){
      sh[0]='H';sh[1]='e';sh[2]='l';sh[3]
    }
    else{
      sh[0]='B';sh[1]='y';sh[2]='e';sh[3
    }
   SetEvent(hEvent1);
   counter++;
  }
}
```

evd11.cpp

```
#include <windows.h>
#include <process.h>
#include <stdio.h>
#pragma comment(lib, "libevd1" )
extern __declspec(dllimport)
char sh[6];
HANDLE hEvent1, hEvent2;
int main(){
  int counter = 0;
  hEvent1 = OpenEvent(EVENT_ALL_ACCESS, FALSE,
                                          "MyTestEvent1");
  hEvent2 = OpenEvent(EVENT_ALL_ACCESS, FALSE,
                                          "MyTestEvent2");
  while (1) {
    WaitForSingleObject(hEvent2, INFINITE);
     if (counter % 2) {
       sh[0] = 'H'; sh[1] = 'e'; sh[2] = 'l'; sh[3] = 'l'; sh[4] = 'o'; sh[5] = '\0';
     }
     else {
       sh[0] = 'B'; sh[1] = 'y'; sh[2] = 'e'; sh[3] = '_'; sh[4] = 'u'; sh[5] = '\0';
     }
     SetEvent(hEvent1);
     counter++;
     Sleep(100);
  }
}
```

Упражнение:

• синхронизируйте потоки разных процессов с помощью мьютексов и семафоров.