Файлы

```
HANDLE WINAPI CreateFile(
      LPCTSTR               lpFileName,
      DWORD                 dwDesiredAccess,
      DWORD                 dwShareMode,                    // 0
      LPSECURITY_ATTRIBUTES lpSecurityAttributes,           // 0
      DWORD                 dwCreationDisposition,
      DWORD                 dwFlagsAndAttributes,           // 0
   HANDLE          hTemplateFile                    // 0
);
```

==========

lpFileName: MAX_PATH (260) – WinDefs.h

dwDesiredAccess GENERIC_READ, GENERIC_WRITE

dwShareMode FILE_SHARE_READ, FILE_SHARE_WRITE

dwCreationDisposition

CREATE_ALWAYS

CREATE_NEW

OPEN_ALWAYS

OPEN_EXISTING

Файлы

```
BOOL ReadFile(
      HANDLE hFile,     // Дескриптор файлу (повертається функцією CreateFile)
      LPVOID lpBuffer,  // Буфер для введених даних
      DWORD nNumberOfBytesToRead, //Кількість байтів для читання
      LPDWORD lpNumberOfBytesRead,  // Кількість байтів, які були  прочитані
      LPOVERLAPPED lpOverlapped   // За звичай 0
);
BOOL WriteFile(
      HANDLE hFile,     // Дескриптор файлу (повертається   функцією CreateFile)
      LPVOID lpBuffer,  // Буфер для введених даних
      DWORD nNumberOfBytesToWrite, //Кількість байтів для введення
      LPDWORD lpNumberOfBytes Written, // Кількість байтів, які були  записані
      LPOVERLAPPED lpOverlapped   // За звичай 0
);
```

Функції повертають:

TRUE при успішному завершенні;

FALSE - в разі помилки введення–виведення.

Файлы

Функції для пошуку файлів та каталогів:

Пошук першого файлу, який задовольняє заданій масці:

```
HANDLE FindFirstFile(
        LPCTSTR lpFileName,             // Ім'я маски, наприклад -    "*.txt"
        LPWIN32_FIND_DATA lpFindFileData    //Адреса структури з  інформацією
);
```

Якщо функція повертає INVALID_HANDLE_VALUE, то немає файлів, які задовольняють масці; якщо інше значення, то пошук було проведено успішно.

Структура:

```
typedef struct _WIN32_FIND_DATA {
DWORD dwFileAttributes;                 // Атрибути файлу
        FILETIME ftCreationTime;// Час створення файлу
        FILETIME ftLastAccessTime;     // Час останнього доступу до файлу
        FILETIME ftLastWriteTime;       // Час останньої модифікації
        DWORD    nFileSizeHigh;  // Розмір файлу (старша частина)
        DWORD    nFileSizeLow;  // Розмір файлу (молодша частина)
        DWORD    dwReserved0;                   // Резерв
        DWORD    dwReserved1;                   // Резерв
        TCHAR    cFileName[ MAX_PATH ];   // Ім'я знайденого файлу
        TCHAR    cAlternateFileName[14];            // Коротке ім'я файлу
} WIN32_FIND_DATA;
```

Файлы

Пошук наступного файлу, який задовольняє масці:

```
BOOL FindNextFile(
      HANDLE hFindFile,                  // Дескриптор файлу
      LPWIN32_FIND_DATA lpFindFileData      //Адреса структури з  інформацією
);
```

Функція повертає TRUE, якщо знайдено наступний файл і FALSE, якщо файлу не знайдено.

Якщо пошук файла по масці завершено необхідно закрити дескриптор:

```
BOOL FindClose( HANDLE hFindFile );
```

де hFindFile - дескриптор, який повертає функція FindFirstFile.

Проверка целостности

Пусть есть программа Add_Crc.exe, которая добавляет в конец проверяемого файла lab1.dll его CRC (будет рассмотрена на практическом занятии).

Пусть ее месторасположение: C:\Study\

Путь к проверяемому файлу: D:\Temp\

Содержимое командного файла Test.bat:

%1Add_Crc.exe %2

Запуск .bat файла из командной строки:

Test.bat C:\Study\ D:\Temp\lab1.dll

аналогичен следующему запуску Add_Crc.exe из командной строки:

C:\Study\Add_Crc.exe D:\Temp\lab1.dll

(решение проблемы путь с пробелами - “его задание в двойных кавычках”)

dllmain.cpp

```
#include "stdafx.h"
#include "tchar.h"
#include <windows.h>



bool CheckCRC(HMODULE hMod)
{    DWORD crc=0,CRCtemplate;
     DWORD High,Low, data,real;
     TCHAR LibName[MAX_PATH];
     HANDLE  hFile;
     GetModuleFileName(hMod,LibName,MAX_PATH);

     hFile=CreateFile(LibName,GENERIC_READ,FILE_SHARE_READ,0,OPEN_EXISTING,0,0);
     if(hFile==INVALID_HANDLE_VALUE)
     {
       _tprintf(_T("DLL not found during CheckCRC!"));  return 0;
     }
```

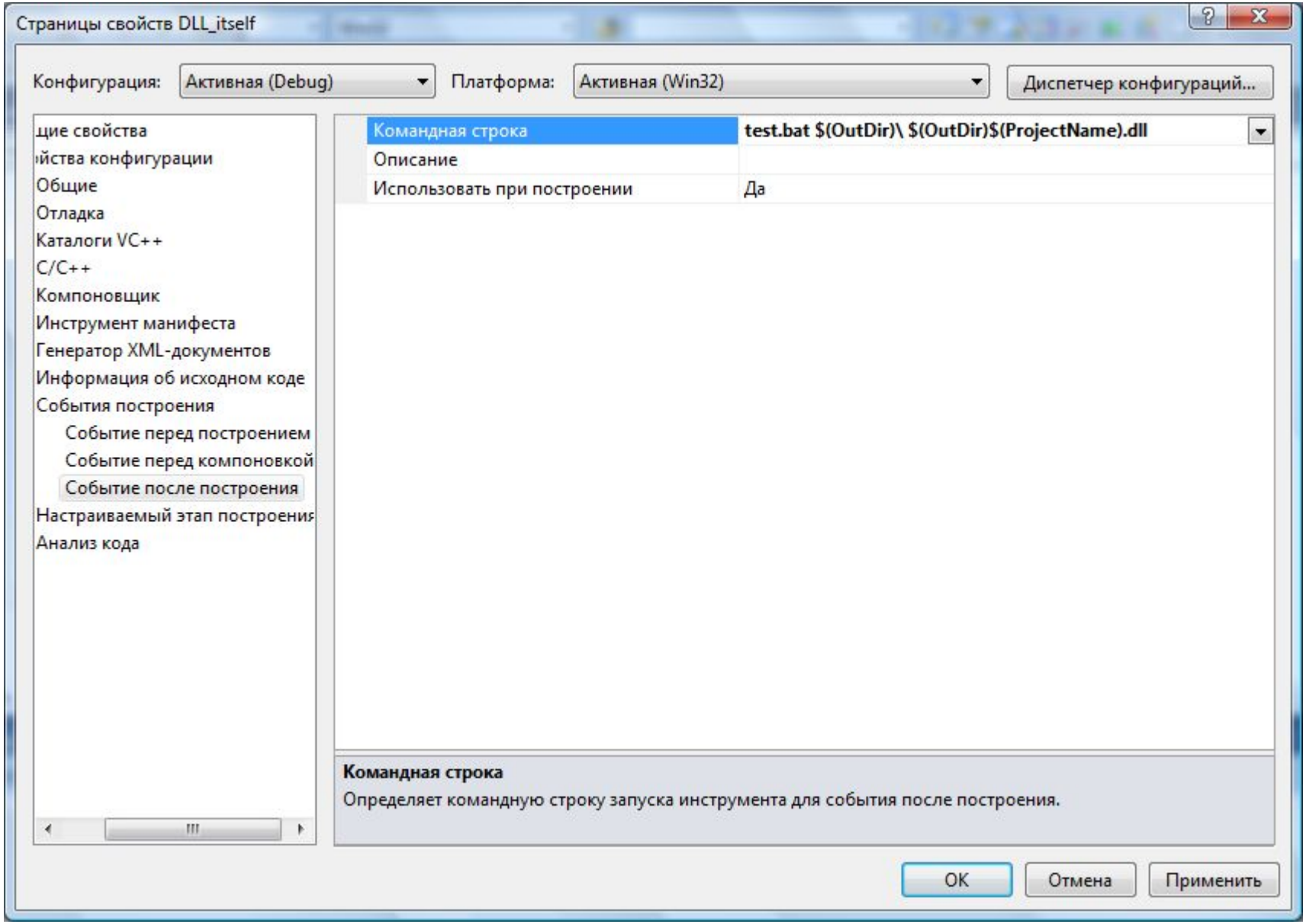
dllmain.cpp

```
// продолжение предыдущего слайда
Low=GetFileSize(hFile,&High);

int counter=(Low-4)/4;
int rem=Low%4;

for(int i=0;i<counter;i++)
{ReadFile(hFile,&data,4,&real,0);
  crc=(crc+data)%0xffff;
}
ReadFile(hFile,&data,rem,&real,0);
crc=(crc+data)%0xffff;


ReadFile(hFile,&CRCtemplate,4,&real,0);
CloseHandle(hFile);
if (CRCtemplate = = crc)
  return true;
else
  return false;
}
```

Файл dllmain.cpp. Точка входа в ДЛЛ.

```
// продолжение предыдущего слайда
BOOL APIENTRY DllMain( HMODULE hModule,   DWORD  ul_reason_for_call,   LPVOID lpReserved  )
{
    switch (ul_reason_for_call)
    {
      case DLL_PROCESS_ATTACH:
                        if ( !CheckCRC(hModule) )
                        {  _tprintf(_T("CheckCRC returns FALSE!\n"));
                           return false;
                        }
                        else
                        {  _tprintf(_T("CheckCRC returns TRUE!\n"));
                           return true;
                        }
      case DLL_THREAD_ATTACH:
      case DLL_THREAD_DETACH:
      case DLL_PROCESS_DETACH:
      break;
}
return TRUE;
}
```

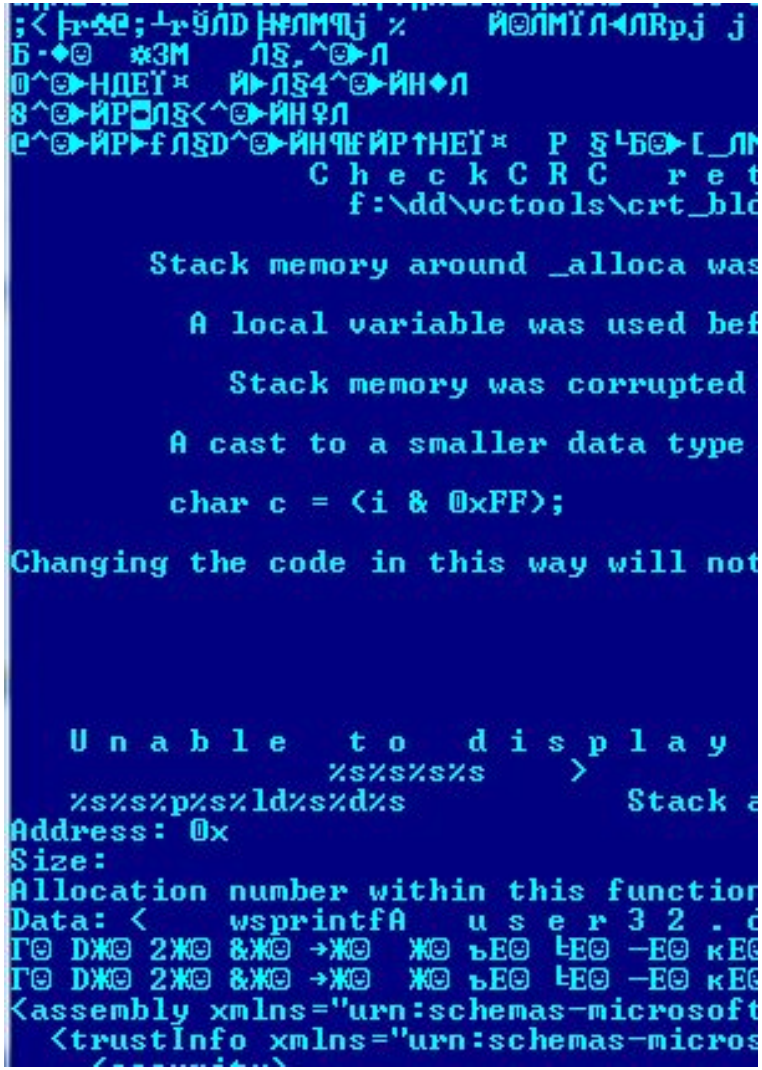
СРЕДА VISUAL STUDIO И КОМАНДНЫЕ ФАЙЛЫ



Программа, загружающая DLL в динамическом режиме

```
……..
int _tmain(int argc, _TCHAR* argv[])
{

    // If DLL was substituted LoadLibrary returns NULL

    hLib=LoadLibrary(_T("DLL_itself.dll"));
    if(hLib==NULL)
    {
      _tprintf(_T("No Library Loaded\n"));
      _getch();
      return -1;
    }

    ……
}
```

Просмотр кода .dll



Лекция 8

Процессы. Межпроцессное взаимодействие.

Процессы

- Создание и завершение процессов;
- Дополнительные функции для работы с процессами;
- Взаимодействие между процессами (IPC - Inter Process Communication);
- Объекты синхронизации. Критические секции и способы их реализации.

Процессы

Процесс – это приложение (программа), которая выполняется процессором.

При создании процессу выделяется адресное пространство, где хранятся его данные.

Программный код, соответствующий данной программе выполняется мастер-потоком процесса.

Процессы

Для каждого процесса в системной области памяти (в области ядра) хранится следующая информация:

- объект ядра типа Process (структура данных с информацией);
- объект ядра типа Thread;
- контекст потока: набор данных для приостановления и продолжения работы (состояние системных регистров (EIP - адрес следующей команды, начиная с которой начнется выполнение прерванного потока). Переключение контекста имеет большие накладные расходы по времени;
- класс приоритет процесса IDLE_PRIORITY_CLASS, **NORMAL_**, HIGH_, REALTIME_... Задается с помощью функции

BOOL SetPriorityClass(HANDLE hProcess, DWORD dwPriorityClass);

(кванты – потокам -> приоритеты потоков 0 – min, 31- max)

- состояние процесса(signaled - свободный , nonsignaled - занятый);
- атрибуты безопасности;
- используемые ресурсы (файлы, библиотеки, окна,..);
- переменные окружения;
- …

Wait -функции

Wait –функции проверяют состояние(я) указанного(ых) объекта(ов). Для некоторых объектов ядра меняют состояние.

```
DWORD WaitForSingleObject(
    HANDLE      hHandle,
    DWORD       dwMilliseconds              //INFINITE
);
```

==========

```
DWORD WaitForMultipleObjects(
    DWORD       nCount,
    const HANDLE*   lpHandles,
    BOOL        bWaitAll,           // TRUE – for all
    DWORD       dwMilliseconds
);
```

Пример запуска процесса

```
TCHAR CommandLine [MAX_PATH] = _T("notepad.exe");
STARTUPINFO StartupInfo = {sizeof (StartupInfo)}; //структура с полями
PROCESS_INFORMATION ProcessInformation; //структура с полями:
                    // hProcess,  hThread,  dwProcessId,  dwThreadId

BOOL b =  CreateProcess(
  NULL,
  CommandLine,
  0,                //LPSECURITY_ATTRIBUTES lpProcessAttributes,
  0,          //LPSECURITY_ATTRIBUTES lpThreadAttributes,
  FALSE,        //BOOL bInheritHandles,
  0,          //DWORD dwCreationFlags,
  0,          //LPVOID lpEnvironment,
  0,          //LPCTSTR lpCurrentDirectory,
  &StartupInfo,
  &ProcessInformation
);
…
WaitForSingleObject( ProcessInformation.hProcess, INFINITE),
CloseHandle(ProcessInformation. hProcess);
```

Действия по завершению процесса

1. Все ресурсы, которые выделены процессу (окна, кучи...) освобождаются.
2. Все потоки, созданные процессом, в том числе первоначальный, отмечаются как свободные.
3. Все объекты ядра, которые использовались процессом, закрываются. Если DLL, то DLL_PROCESS_DETACH.
4. Освобождается память, которая занята кодом и данными.
5. Устанавливается код завершения процесса (return 0)
6. Процесс переходит в состояние Свободный
7. Входная функция первичного потока возвратила управление

Процессы. Пример

```
#include "stdafx.h"
#include <windows.h>
#include <stdio.h>      // программа запускает notepad.exe, ждет пока пользователь его не закроет,
#include <conio.h>      //  потом ищет в текущем каталоге текстовые файлы, которые были созданы
#include <tchar.h>      //  только что, и архивирует их в архив test.rar
#include <locale.h>
int _tmain(int argc, _TCHAR* argv[]) {
SYSTEMTIME        st;      // structure  WORD wYear;  WORD wMonth;  WORD wDayOfWeek;
FILETIME ft;    // structure represents the number of 100-nanosecond since January 1, 1601
                //  DWORD dwLowDateTime;  DWORD dwHighDateTime;
_tsetlocale(LC_ALL,_T("Russian"));
GetLocalTime(&st);   //the date and time of day for your time zone.
SystemTimeToFileTime(&st,&ft);
PROCESS_INFORMATION pi; // HANDLE hProcess;  HANDLE hThread;  DWORD dwProcessId; DWORD dwThreadId;
STARTUPINFO si;        //specify the window station and appearance of the main window for the new
      process.
memset(&si,0,sizeof(si));
si.cb=sizeof(si);
unsigned _int64 MinTime= ((unsigned _int64)ft.dwHighDateTime<<32)|ft.dwLowDateTime;
TCHAR EditorName[]=_T("notepad.exe");
BOOL b=CreateProcess(0,EditorName,0,0,false,0,0,0,&si,&pi);
```

Процессы. Пример

```
 if (!b) { printf("Error\n"); return -1; }                    // продолжение предыдущего слайда
WaitForSingleObject(pi.hProcess,INFINITE);
CloseHandle(pi.hThread);CloseHandle(pi.hProcess);
HANDLE hFile;
TCHAR n[]=_T("arch1.bat");
hFile=CreateFile(n,GENERIC_WRITE,FILE_SHARE_READ,0,CREATE_ALWAYS,0,0);
if (!hFile) { _tprintf(_T(".bat file was not created")); return -1; }
DWORD len;
char templ[]="\"E:\\Program Files\\WinRAR\\winrar.exe\" a test ";
            // заменить на путь на Вашем комп
char Converted[MAX_PATH];
WriteFile(hFile,templ,strlen(templ),&len,0);
WIN32_FIND_DATA FindFileData; //searches a directory for a file with a name
HANDLE h=FindFirstFile(_T("*.txt"),&FindFileData  );  // that matches a specific name
if (h==INVALID_HANDLE_VALUE)
{
    printf("File not found");
    return -1;
}
```

Процессы. Пример

```
while (1){ // продолжение предыдущего слайда
 FileTimeToLocalFileTime(&FindFileData.ftCreationTime,&ft);
unsigned _int64 CreateTime= ((unsigned _int64)ft.dwHighDateTime<<32)|ft.dwLowDateTime;
if (CreateTime>MinTime){
   _tprintf(_T("%s\n"),FindFileData.cFileName);
   #ifdef UNICODE
     WideCharToMultiByte(CP_OEMCP, 0, FindFileData.cFileName,-1, Converted, sizeof(Converted), 0,0);
   #else
     CharToOem(FindFileData.cFileName, Converted);
   #endif
   WriteFile(hFile,Converted,strlen(Converted),&len,0);
   WriteFile(hFile," ",strlen(" "),&len,0);
}
b=FindNextFile(h,&FindFileData);
if (!b)break;
}
FindClose(hFile);
CloseHandle(hFile);
b=CreateProcess(0,n,0,0,false,0,0,0,&si,&pi);
if (!b){_tprintf(_T("Error\n"));return 1;}
WaitForSingleObject(pi.hProcess,INFINITE);
CloseHandle(pi.hThread); CloseHandle(pi.hProcess); _getch();       return 0;     }
```

Межпроцессное взаимодействие. Inter Process Communication

3 проблемы для решения:

- Использование общих данных разными процессами (FileMapping, Environment Variables, командная строка)
- Корректное использование общих данных (запись + чтение)
- Обеспечение необходимого порядка выполнения операций (спулер принтера)

Создание, изменение и использование переменных окружения

DWORD **GetEnvironmentVariable**(LPCTSTR *lpName*, LPTSTR *lpBuffer*, DWORD *nSize*);

BOOL **SetEnvironmentVariable**(LPCTSTR *lpName*, LPCTSTR *lpValue*);

Пример:

```
#include "windows.h"
// Если запустить без аргументов ком. строки  MyVersion = Trial
int _tmain(int argc,_TCHAR* argv[])
{
    TCHAR tcTrial [] = _T("Trial");
    TCHAR tcWork [] = _T("Work");
    TCHAR *value = ( argc == 1)? tcTrial : tcWork;
    BOOL b = SetEnvironmentVariable( _T("MyVersion"), value);
    if (b)
    {
      TCHAR Buffer[4096];
      GetEnvironmentVariable( _T("MyVersion"), Buffer, 4096);
      _tprintf (_T("MyVersion = %s\n"), Buffer);
    }
    ...
}
```

Использование общей памяти. Гонки (Race Conditions)

x=2;

Процесс1	**Процесс2**

x++;
 	
==========
mov eax, [x];
dec eax
mov [x], eax
=========
	x++;
	
	==========
	mov eax, [x];
	dec eax
	mov [x], eax
	=========

Критическая секция. Определение

Определение.

Участок кода называется критическим, если его выполнение одновременно должен делать только один процесс (эксклюзивный доступ, критическая секция).

Когда создается?

Доступ к общему ресурсу (память, файлы,...) в режиме модификации несколькими процессами.

Критическая секция. Требования

1. Никакие 2 процесса не могут одновременно войти в КС
2. Никаких предположений о скорости выполнения и количестве процессов сделать нельзя
3. Если процесс не выполняет КС, он не должен блокировать выполнение других процессов
4. Если процесс выполняет КС, не должны блокироваться процессы, которые не выполняют КС.
5. Никакой процесс не должен ждать бесконечноо долго входа в КС

Способы реализации КС. 1 способ

Использование общей переменной для блокирования

Процес 0	Процес 1
int block =0;	

while (1){
while (block);
block = 1;
CS
block = 0;
NCS
}
	while (1){
	while (block);
	block = 1;
	CS
	block = 0;
	NCS
	}

Постоянно требует времени процессора для проверки состояния переменной вместо засыпания

Способы реализации КС. 2 способ

Использование общей сменной для блокирования с поочередным выполнением процессов.

Процес 0	**Процес 1**
int turn = 0	

while (1){
NCS
int process = 0;
while (turn != process);
CS
turn = 1- process;
NCS
}
	while (1){
	NCS
	int process = 1;
	while (turn != process);
	CS
	turn = 1- process;
	NCS
	}