Process Management and IPC

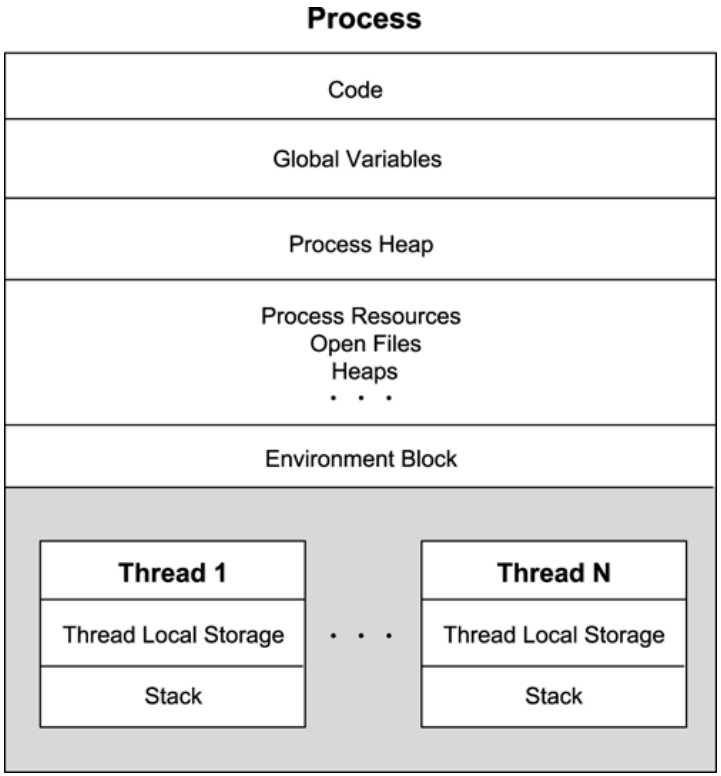
in Windows

Κυριάκος Ιωάννου
Ιάκωβος Πιθαράς

Δρ. Δημήτρης Ζεϊναλιπούρ
Δρ. Χρυσόστομος Χρυσοστόμου

Processes and Theads in Windows

- Διεργασίες στο λειτουργικό σύστημα Windows :
  - Διεργασία
    - Νήμα



Process ID

- Για να δούμε την ταυτότητα της παρούσας διεργασίας καλούμε τη συνάρτηση
  DWORD WINAPI GetCurrentProcessId()

```
#include <stdio.h>
#include <stdlib.h>
#include <Windows.h>
int main(int argc, char *argv[])
{
	printf("my id is : %d \n",GetCurrentProcessId());
	system("PAUSE");
	return 0;
}
```

Δημιουργία διεργασίας

- Για να δημιουργήσουμε νέα διεργασία στα Windows υπάρχουν αρκετοί τρόποι.
- Η βασικότερη συνάρτηση είναι η CreateProcess.
- BOOL CreateProcess (LPCTSTR lpApplicationName,
  LPTSTR lpCommandLine,
  LPSECURITY_ATTRIBUTES lpsaProcess,
  LPSECURITY_ATTRIBUTES lpsaThread,
  BOOL bInheritHandles,
  DWORD dwCreationFlags,
  LPVOID lpEnvironment,
  LPCTSTR lpCurDir,
  LPSTARTUPINFO lpStartupInfo,
  LPPROCESS_INFORMATION lpProcInfo)

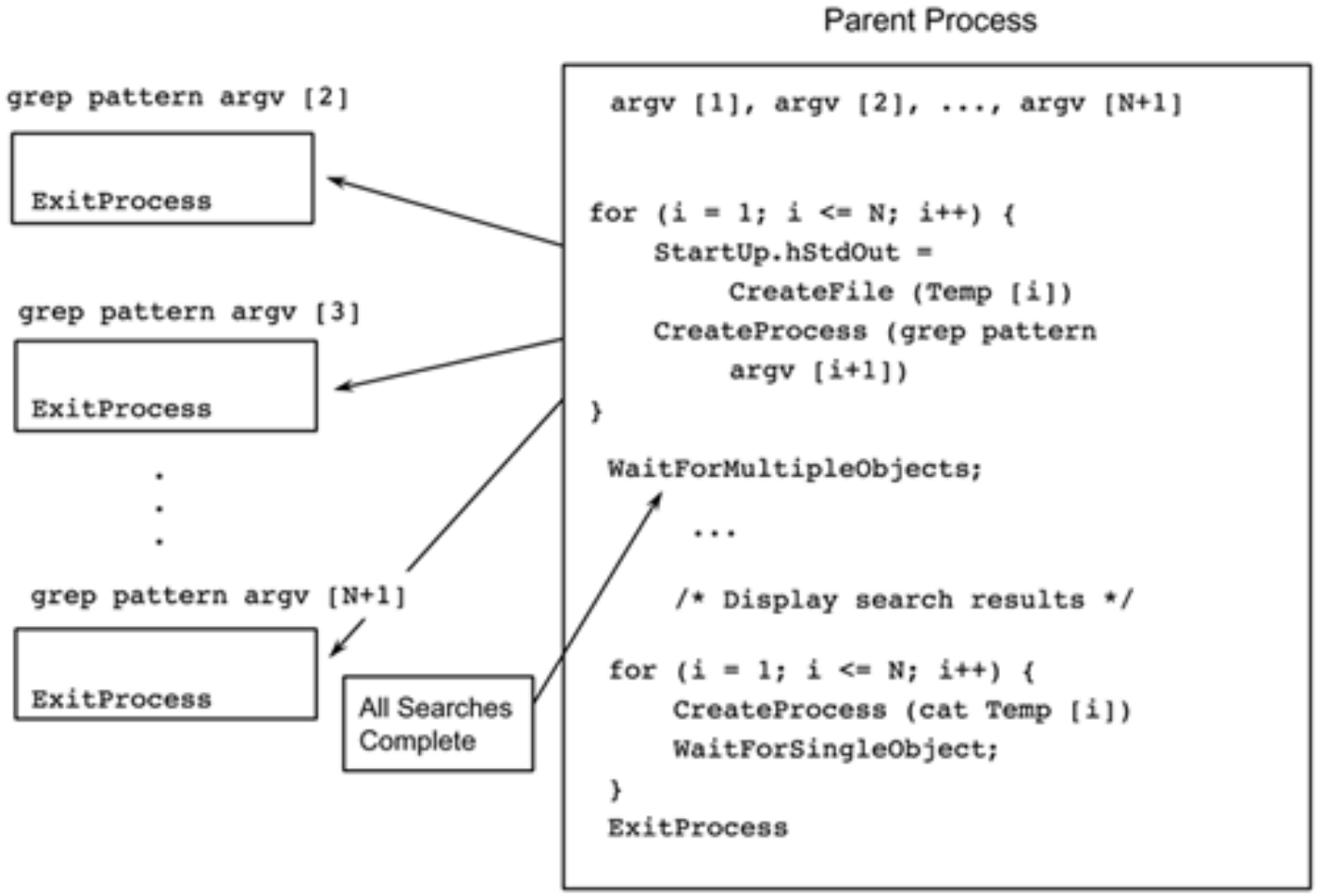
Δημιουργία διεργασίας(παράδειγμα)

```
/*   Chapter 6. grepMP. */
/*   Multiple process version of grep command. */

#include "EvryThng.h"
int _tmain (DWORD argc, LPTSTR argv [])
/* Create a separate process to search each file on the
   command line. Each process is given a temporary file,
   in the current directory, to receive the results. */
{
   HANDLE hTempFile;
   SECURITY_ATTRIBUTES StdOutSA = /* SA for inheritable handle. */
         {sizeof (SECURITY_ATTRIBUTES), NULL, TRUE};
   TCHAR CommandLine [MAX_PATH + 100];
   STARTUPINFO StartUpSearch, StartUp;
   PROCESS_INFORMATION ProcessInfo;
   DWORD iProc, ExCode;
   HANDLE *hProc; /* Pointer to an array of proc handles. */
   typedef struct {TCHAR TempFile [MAX_PATH];} PROCFILE;
   PROCFILE *ProcFile; /* Pointer to array of temp file names. */
   GetStartupInfo (&StartUpSearch);
   GetStartupInfo (&StartUp);
   ProcFile = malloc ((argc - 2) * sizeof (PROCFILE));
   hProc = malloc ((argc - 2) * sizeof (HANDLE));

   /* Create a separate "grep" process for each file. */
   for (iProc = 0; iProc < argc - 2; iProc++) {
      _stprintf (CommandLine, _T ("%s%s %s"),
         _T ("grep "), argv [1], argv [iProc + 2]);
      GetTempFileName (_T ("."), _T ("gtm"), 0,
            ProcFile [iProc].TempFile); /* For search results. */
      hTempFile = /* This handle is inheritable */
         CreateFile (ProcFile [iProc].TempFile,
            GENERIC_WRITE,
            FILE_SHARE_READ | FILE_SHARE_WRITE, &StdOutSA,
            CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
      StartUpSearch.dwFlags = STARTF_USESTDHANDLES;
      StartUpSearch.hStdOutput = hTempFile;
      StartUpSearch.hStdError = hTempFile;
      StartUpSearch.hStdInput = GetStdHandle (STD_INPUT_HANDLE);
   /* Create a process to execute the command line. */
      CreateProcess (NULL, CommandLine, NULL, NULL,
            TRUE, 0, NULL, NULL, &StartUpSearch, &ProcessInfo);
      /* Close unwanted handles. */
      CloseHandle (hTempFile); CloseHandle (ProcessInfo.hThread);
      hProc [iProc] = ProcessInfo.hProcess;
   }
   /* Processes are all running. Wait for them to complete. */
   for (iProc = 0; iProc < argc - 2; iProc += MAXIMUM_WAIT_OBJECTS)
      WaitForMultipleObjects ( /* Allows a large # of processes */
            min (MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc),
            &hProc [iProc], TRUE, INFINITE);
   /* Result files sent to std output using "cat." */
   for (iProc = 0; iProc < argc - 2; iProc++) {
      if (GetExitCodeProcess(hProc [iProc], &ExCode) && ExCode==0) {
         /* Pattern was detected -- List results. */
         if (argc > 3) _tprintf (_T ("%s:\n"), argv [iProc + 2]);
         fflush (stdout); /* Multiple processes use stdout. */
         _stprintf (CommandLine, _T ("%s%s"),
               _T ("cat "), ProcFile [iProc].TempFile);
         CreateProcess (NULL, CommandLine, NULL, NULL,
               TRUE, 0, NULL, NULL, &StartUp, &ProcessInfo);
         WaitForSingleObject (ProcessInfo.hProcess, INFINITE);
         CloseHandle (ProcessInfo.hProcess);
         CloseHandle (ProcessInfo.hThread);
      }
      CloseHandle (hProc [iProc]);
      DeleteFile (ProcFile [iProc].TempFile);
   }
   free (ProcFile);
   free (hProc);
   return 0;
}
```

Δημιουργία διεργασίας(παράδειγμα)



Δημιουργία διεργασίας

- Η διεργασία παιδί μπορεί να κληρονομήσει αρκετές ιδιότητες και πόρους από τον πατέρα της αλλά και να εμποδιστεί από το να τις κληρονομήσει. Γενικά μπορεί να κληρονομήσει τα ακόλουθα:
  - Ανοικτά χειριστήρια που επιστρέφονται από τη συνάρτηση **CreateFile**. Αυτά περιλαμβάνουν χειριστήρια σε αρχεία, input buffers κονσόλας , buffers οθόνης της κονσόλας, named pipes, serial communication devices και mailslots.
  - Ανοικτά χειριστήρια σε διεργασίες, νήματα, mutex, event, semaphore, named-pipe, anonymous-pipe και file-mapping αντικείμενα.
  - Μεταβλητές περιβάλλοντος.
  - Τον τρέχων κατάλογο.
  - Την κονσόλα, εκτός εάν η διεργασία αποσυνδέεται ή εάν μία νέα κονσόλα δημιουργείται. Η διεργασία παιδί, μπορεί επίσης να κληρονομήσει τα σταθερά χειριστήρια του πατέρα της, όπως επίσης και να έχει πρόσβαση στο input buffer και στο active screen buffer.
  - Το error mode, όπως ορίζεται από τη συνάρτηση **SetErrorMode**.
  - Το process affinity mask (συσχετισμός νήματος με επεξεργαστή – σε multiprocessor συστήματα) .
  - Το συσχετισμό με ένα job.

Δημιουργία διεργασίας

- Η διεργασία δεν μπορεί να κληρονομήσει τα ακόλουθα:
  - Την κλάση προτεραιότητας.
  - Χειριστήρια που επιστρέφονται από τις συναρτήσεις **LocalAlloc**, **GlobalAlloc**, **HeapCreate** και **HeapAlloc**.
  - Ψευδοχειριστήρια, όπως τα χειριστήρια που επιστρέφονται από τις συναρτήσεις **GetCurrentProcess** ή **GetCurrentThread** . Τα χειριστήρια αυτά ισχύουν μόνο για την καλούσα διεργασία.
  - Χειριστήρια σε DLL module που επιστρέφονται από την συνάρτηση **LoadLibrary**.
  - Χειριστήρια σε GDI ή USER , όπως **HBITMAP** ή **HMENU**.

Wait command

Η διεργασία πατέρας χρησιμοποιεί τις διεργασίες WaitForMultipleObjects και WaitForSingleObject, για να περιμένει τις διεργασίες παιδιά να τερματίσουν πριν να συνεχίσει η ίδια.

- DWORD WINAPI WaitForMultipleObjects(
  DWORD *nCount*, /*αριθμός διεργασιών που περιμένει*/
  const HANDLE* *lpHandles*, /*χειριστήρια στις διεργασίες*/
  BOOL *bWaitAll*,
  DWORD *dwMilliseconds* /*χρόνος αναμονής των διεργασιών*/);

Επιστρέφει τον κωδικό του event που προκάλεσε τον τερματισμό της συγκεκριμένη διεργασίας

- DWORD WINAPI WaitForSingleObject(
  HANDLE *hHandle*, /*χειριστήριο στη διεργασία*/
  DWORD *dwMilliseconds*);

Επιστρέφει τον κωδικό του event που προκάλεσε τον τερματισμό της συγκεκριμένης διεργασίας

Τερματίζοντας μια διεργασία

- Όταν τερματίζεται μια διεργασία προκαλούνται τα εξής:
  - Όλα τα εναπομείναντα νήματα σημειώνονται για τερματισμό.
  - Ελευθερώνονται όλοι οι πόροι που δεσμεύτηκαν από τη διεργασία.
  - Κλείνονται όλα τα αντικείμενα του πυρήνα.
  - Ο κωδικός της διεργασίας αφαιρείται από τη μνήμη.
  - Γράφεται ο κωδικός εξόδου της διεργασίας.
  - Σηματοδοτείται το αντικείμενο της διεργασίας.
- Η συνάρτηση **GetExitCodeProcess** επιστρέφει την κατάσταση τερματισμού της διεργασίας.
- Δεν τερματίζονται διεργασίες που δημιουργήθηκαν από την διεργασία που τερματίστηκε.
- Μια διεργασία τερματίζεται όταν:
  - Ένα νήμα της διεργασίας καλέσει την συνάρτηση **ExitProcess**
  - Τερματιστεί το τελευταίο νήμα της διεργασίας.
  - Οποιοδήποτε νήμα καλέσει την **TerminateProcess** με χειριστήριο στη διεργασία.
  - Για διεργασίες κονσόλας,ο default χειριστής ελέγχου κονσόλας καλεί την **ExitProcess** όταν η κονσόλα πάρει το CTRL+C or CTRL+BREAK.
  - Ο χρήστης κλείσει το σύστημα ή αποσυνδεθεί.

Environment Variables

Κάθε διεργασία έχει ένα block περιβάλλοντος, το οποίο περιέχει ένα σετ από μεταβλητές περιβάλλοντος καθώς και τις τιμές τους. Μια διεργασία παιδί, κληρονομεί ,εξ'ορισμού, τις μεταβλητές περιβάλλοντος του πατέρα της. Παρόλα αυτά είναι δυνατό, να οριστεί νέο περιβάλλον για τη διεργασία παιδί, με το να δημιουργηθεί νέο block περιβάλλοντος και να περαστεί ένας δείκτης σε αυτό, σαν παράμετρος της συνάρτησης CreareProcess.

Environment Variables

- Οι μεταβλητές περιβάλλοντος, μέσα στο μπλόκ περιβάλλοντος, έχουν την ακόλουθη μορφή:
  Var1=*Value1*\0
  Var2=*Value2*\0
  Var3=*Value3*\0
  ...
  VarN=*ValueN*\0\0

Environment Variables

- Με τη συνάρτηση LPTCH WINAPI GetEnvironmentStrings(void); επιστρέφεται ένας δείκτης στο μπλόκ περιβάλλοντος της καλούσας διεργασίας που επιτρέπει μόνο ανάγνωση στις μεταβλητές.

- Με τη συνάρτηση BOOL WINAPI SetEnvironmentVariable(LPCTSTR *lpName*, LPCTSTR *lpValue*); επιτρέπεται η αλλαγή των μεταβλητών.

Ορισμός - IPC

- IPC είναι το ακρωνύμιο της φράσης Inter-Process Communication. Βασίζεται κυρίως στους μηχανισμούς και τεχνικές της διαδιεργασιακής επικοινωνίας.

- Επίσης μπορούμε να το ορίσουμε σαν την ικανότητα που έχει ένα λειτουργικό σύστημα έτσι ώστε να μπορεί μια διεργασία να επικοινωνίση με μια άλλη διεργασία. Η διεγργασία μπορεί να είναι στον ίδιο υπολογιστή ή μπορεί να είναι συνδεδεμένοι στο δίκτυο.

Μηχανισμοί IPC

Βασικοί μηχανισμοί για Windows :

- Shared Memory
- Pipes
- WinSock
- Mailslot
- OLE

Για την λειτουργία των IPC χρησιμοποιούνται κάποιοι μηχανισμοί
συγχρονισμού :

- Mutex
- Semaphore
- Events

Inter-process Synchronization

HANDLE WINAPI CreateEvent(LPSECURITY_ATTRIBUTES *lpEventAttributes*,
BOOL *bManualReset*,
BOOL *bInitialState*,
LPCTSTR *lpName*);

HANDLE WINAPI CreateMutex(LPSECURITY_ATTRIBUTES *lpMutexAttributes*,
BOOL *bInitialOwner*,
LPCTSTR *lpName*);

HANDLE WINAPI CreateSemaphore(LPSECURITY_ATTRIBUTES *lpSemaphoreAttributes*,
LONG *lInitialCount*,
LONG *lMaximumCount*,
LPCTSTR *lpName*);

Shared Memory

Γνωστό και σαν File Mapping

- *Επιτρέπει στην επικοινωνία διεργασιών χρησιμοποιώντας το περιεχόμενο ενός κειμένου στην μνήμη.*

- *Η διεργασία μπορεί να χρησιμοποιήσει ένα απλό διαχειριστή αρχείων και να τροποποιήσει το περιεχόμενο ενός αρχείου.*

- *Δύο ή περισσότερες διεργασίες με το ίδιο file mapping διαβάζουν από το κοινό αρχείο.*

•Μπορεί να χρησιμοποιηθεί απο διεργασίες στον ίδιο υπολογιστή μόνο.

•Είναι πολύ εύκολη η εγκαθίδρυση file mapping

Shared Memory

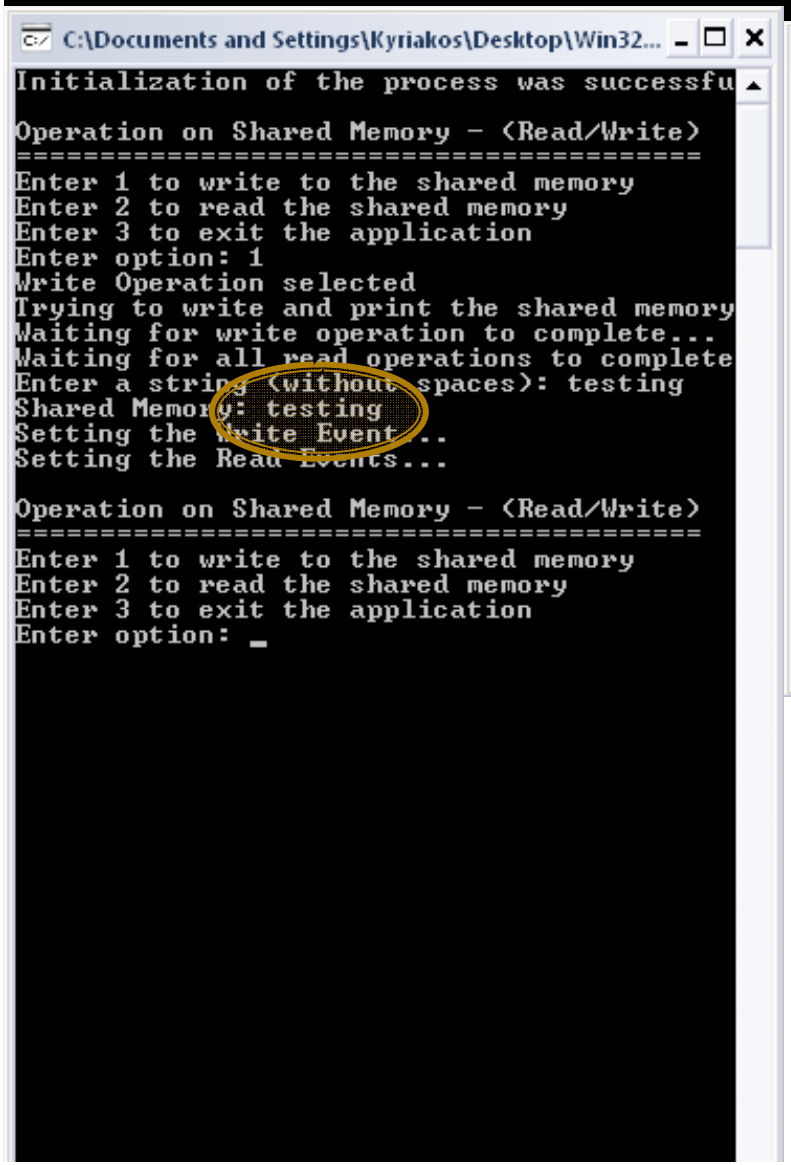
Οι κύριες εντολές απο το Win32 API που μπορούμε να διαχειριστούμε το Shared Memory είναι:

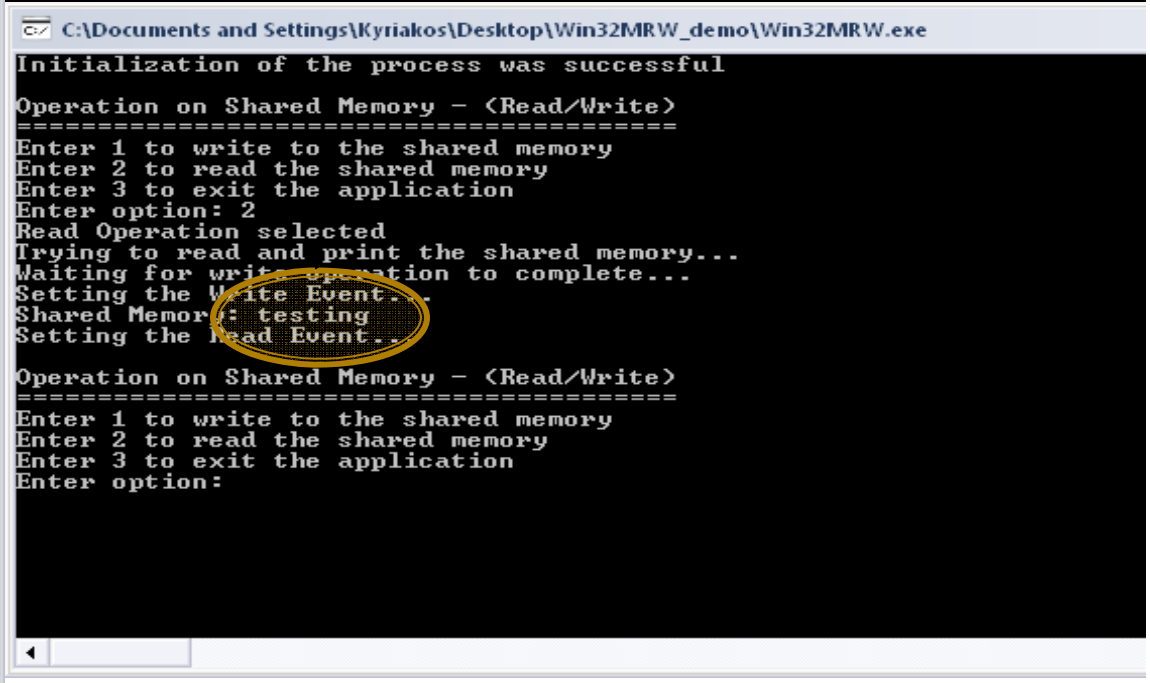
- CreateFileMapping()
- MapViewOfFile()
- UnMapViewOfFile()
- CloseHandle()

```
g_hSharedMemory = CreateFileMapping(
HANDLE hFile,                  // handle to file
LPSECURITY_ATTRIBUTES,         // security attribut
PAGE_READWRITE,                // read/write access
O,                             // max. object size
MAX_SH_MEM_SIZE,               // buffer size
g_szShareMemoryName); // name of mapping object
```

```
g_pBuffer = (LPTSTR) MapViewOfFile(
g_hSharedMemory,               // handle to map object
FILE_MAP_ALL_ACCESS,           // read/write perm
O,
O,
MAX_SH_MEM_SIZE);
```

Shared Memory





1. Πρώτα επιλέγουμε read from the shared memory
2. Ο reader περιμένει
3. Στο άλλο παράθυρο κάνουμε write to the shared memory
4. Γράφουμε στην κοινή μνήμη.
5. Ο reader διαβάζει από την κοινή μνήμη

Pipes

Τα pipes λειτουργούν σαν FIFO

Υπάρχουν δύο ειδών Pipes – **Anonymous Pipes** και **Named Pipes**

Ένα anonymous pipe είναι :

- τοπικό, δεν μπορεί να επικοινωνίση με το δίκτυο
- δεν έχει όνομα
- είναι μονής κατεύθυνσης
- χρησιμοποιείται για μεταφορά δεδομένων από μια διεργασία πατέρας στο παιδί της

Named Pipes

- Μπορούν να είναι duplex
- Μπορούν να χρησιμοποιηθούν απο διεργασίες στον ίδιο υπολογιστή αλλά και στο δίκτυο.
- Διαχείρηση σαν κανονικά αρχεία
- Χρησιμοποιούνται κυρίως σαν client-server επικοινωνία
- Δεν είναι special αρχεία όπως τα UNIX

Named Pipes

Server

```
Dim byteCount, i, res, cbnCount As Integer
For i = 0 To BUFFSIZE - 1 'Fill an array of numbers
   Buffer(i) = i Mod 256
Next i
'Wait for a connection, block until a client connects
Label1.Text = "Waiting for client connections"
Me.Refresh()
Do
   res = ConnectNamedPipe(hPipe, 0)
   'Read the data sent by the client over the pipe
   cbnCount = 4
   res = ReadFile(hPipe, byteCount, Len(byteCount), cbnCount,
0)
   If byteCount > BUFFSIZE Then 'Client requested for byteCount
bytes
      byteCount = BUFFSIZE 'but only send up to 20000 bytes
   End If
   'Write the number of bytes requested by the client
   res = WriteFile(hPipe, Buffer, byteCount, cbnCount, 0)
   res = FlushFileBuffers(hPipe)
   'Disconnect the named pipe.
   res = DisconnectNamedPipe(hPipe)
   'Loop until the client makes no more requests for data.
Loop Until byteCount = 0
Label1.Text = "Read or Write completed"
```

Named Pipes

Client

```
Dim i, res, cbRead,numBytes As Integer
Dim bArray() As Byte
Dim temp As String

numBytes = CInt(TextBox1.Text)
If numBytes < 0 Then
   MessageBox.Show("Value must be at least 0.", MsgBoxStyle.OKOnly)
   Exit Sub
End If
If numBytes = 0 Then
   Label1.Visible = True
   Label1.Text = "The connection to the server is disconnected."
   Button1.Visible = False
   TextBox1.Visible = False
   TextBox2.Visible = False
End If
If numBytes > BUFFSIZE Then
   numBytes = BUFFSIZE
End If

ReDim bArray(numBytes) 'Create the return buffer
'Call the CallNamedPipe function to do the transactions
 res = CallNamedPipe(pipeName, numBytes, Len(numBytes), bArray(0), numBytes, cbRead, 30000)
'Wait up to 30 seconds for a response
'Format the data received, and then display the data in the text box
If res > 0 Then
   temp = Format(bArray(0), " 000")
   For i = 1 To cbRead - 1
      If (i Mod 16) = 0 Then temp = temp & vbCrLf
         temp = temp & " " & Format(bArray(i), "000")
   Next i
   TextBox2.Text = temp
Else
MessageBox.Show("Error number " & Err.LastDllError & _
"while trying to call the CallNamedPipe function.", MsgBoxStyle.OKOnly)
```

Create a Named Pipe

1. *Ένα πρόγραμμα χρησιμοποιώντας την συνάρτηση ConnectNamedPipe δημιουργά named pipe.*

2. *Ένωση client προγραμματος με server χρησιμοποιώντας την CallNamedPipe function.*

3. *Εκτέλεση της ReadFile ή της WriteFile function για την επικοινωνία με την pipe.*

4. Καλούμε την *DisconnectNamedPipe για να τερματίσουμε την επικοινωνία.*

5. Καλούμε την *CloseHandle στην συνδεδεμένη named pipe για να τερματίσουμε την σύνδεση.*

WinSock

- Τα Windows Sockets (WinSock) είναι βασισμένα στον τρόπο που είναι υλοποιημένα στα UNIX
- Χρησιμοποιώντας Windows Sockets μπορείς να επικοινωνίσης με οποιαδήποτε σύστημα στο δίκτυο που υποστηρίζει Windows Sockets.
- Είναι το πιο διαδεδομένο πρωτόκολλο διεργασιακής επικοινωνίας. Επίσης μπορεί να χρησιμοποιηθεί στην επικοινωνία με άλλα συστήματα και πλατφόρμες αφού σχεδόν

WinSock - Server

```
#include <winsock2.h>

 ListenSocket = socket(AF_INET,
SOCK_STREAM, IPPROTO_TCP);

nPortNo = atoi(argv[1]);

 ServerAddress.sin_family = AF_INET;
    ServerAddress.sin_addr.s_addr =
INADDR_ANY; //WinSock will supply
address
    ServerAddress.sin_port =
htons(nPortNo);    //comes from
commandline

bind(ListenSocket, (struct sockaddr *)
&ServerAddress,
sizeof(ServerAddress))
```

```
accept(ListenSocket, (struct
sockaddr *) &ClientAddress,
&nClientLength);

 nBytesRecv =
recv(RemoteSocket, szBuffer,
255, 0 );

send(RemoteSocket,
ACK_MESG_RECV ,
strlen(ACK_MESG_RECV), 0);

 closesocket(RemoteSocket);
```

WinSock - Client

```
#include <winsock2.h>

nPortNo = atoi(argv[2]);
Socket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
Server = gethostbyname(argv[1]);
ZeroMemory((char *) &ServerAddress, sizeof(ServerAddress));

ServerAddress.sin_family = AF_INET;
copyMemory((char *)&ServerAddress.sin_addr.s_addr,
                                    (char *)Server->h_addr,
                                    Server->h_length);
ServerAddress.sin_port = htons(nPortNo);

nBytesSent = send(Socket, szBuffer, strlen(szBuffer), 0);
nBytesRecv = recv(Socket, szBuffer, 255, 0 );

closesocket(Socket);
```

Named Pipes VS Sockets

Σε περιβάλλοντα που το δίκτυο LAN είναι πολύ γρήγορο τα Sockets και τα Named Pipes έχουν την ίδια ταχύτητα και απόδοση.

Διαφορές στην ταχύτητα όμως παρουσιάζονται σε αργές συνδέσεις όπως dial-up και DSL.

•Στα named pipes για κάθε read που κάνει ένας user στέλνονται named pipes messages πρίν την έναρξη του read.

•Στα Sockets η μεταφορά δεδομένων έχει λιγότερα overheads και μπορούν να εκμεταλευτούν μηχανισμοί όπως windowing, delayed, acknowledgments απο το TCP/IP. Έτσι έχουμε καλύτερη απόδοση σε αργές συνδέσεις.

Σε τελική ανάλυση τα Sockets είναι καλύτερα σε περιβάλλοντα που η ταχύτητα του δικτύου είναι εμπόδιο. Παρόλα αυτα τα named pipes σε γρήγορα δίκτυα είναι καλύτερα αφου προσφέρουν ευκολία χρήσης και πολλά configuration options.

Mailslot

- Παρέχουν επικοινωνία μονής κατεύθηνσης.
- Μιά διεργασία που δημιουργά mailslot είναι ο mailslot server
- Mailslot clients στέλνουν μηνύματα στο mailslot του server
- Το mailslot αποθηκεύει το μήνυμα μέχρι να διαβαστεί απο τον server

Mailslot

- Μια διεργασία μπορεί να είναι και server και client
- Η χρήση του Mailslot είναι πολύ πιο εύκολη από τα pipes ή τα sockets αλλά είναι πιο περιορισμένη.
- Δεν προσφέρουν επιβεβαίωση ότι τα μηνύματα παραλήφθηκαν εκτός και άν είναι υλοποιημένο απο το πρόγραμμα.
- Είναι καλή επιλογή όταν θέλεις να κάνεις broadcast ένα μήνυμα σε πολλά process ή χρειάζεσαι κατι απλό και γρήγορο.

Mailslot

Η πιο διαδεδομένη χρήση του Mailslot είναι το Messenger Service που υπάρχει στα Windows XP. Το Messenger Service είναι ένας Mailslot server που περιμένει να έρθει ένα μήνυμα. Όταν το μήνυμα φτάσει, εμφανίζεται στην οθόνη.

Mailslot - Create and Write

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

HANDLE hSlot;
LPTSTR lpszSlotName= TEXT("\\\\.\\mailslot\\sample_mailslot");

hSlot = CreateMailslot(
            lpszSlotName,
            0,                                  // maximum message size
            MAILSLOT_WAIT_FOREVER,              // no time-out for operations
            (LPSECURITY_ATTRIBUTES) NULL);      // default security

fResult = WriteFile(
            hSlot,
            Message,
            (DWORD) (lstrlen(lpszMessage)+1)*sizeof(TCHAR),
            &cbWritten,
            (LPOVERLAPPED) NULL);
```

Mailslot - Read

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

HANDLE hSlot;
LPTSTR SlotName = TEXT("\\\\.\\mailslot\\sample_mailslot");

MakeSlot(SlotName);
fResult = GetMailslotInfo(   hSlot, // mailslot handle
                             (LPDWORD) NULL, // no maximum message size
                             &cbMessage, // size of next message
                             &cMessage, // number of messages
                             (LPDWORD) NULL); // no read time-out
fResult = ReadFile(hSlot,
                   lpszBuffer,
                   cbMessage,
                   &cbRead,
                   &ov);

CloseHandle(hEvent);
```

Other IPCs - Clipboard

- Προσωρινή αποθήκευση δεδομένων
- Αποθήκευση σε διάφορες μορφές
  - κείμενο, φωτογραφία, γραφική παράσταση, πίνακας
- Χρησιμοποιήται από όλες σχεδόν τις εφαρμογές
- Πολύ εύκολος τρόπος ανταλλαγής δεδομένων

Other IPCs - OLE

- Τροποποίηση του περιεχομένου μιας διεργασίας χρησιμοποιώντας μια άλλη διεργασία
  - π.χ : Γραφική παράσταση σε αρχείο Word
- Το OLE προετοιμάζει και επιτρέπει επεξεργασίας της γραφικής παράστασης
- Χρήση άλλων εφαρμογών διαμέσου άλλων εφαρμογών

Πηγές:

Papers:

Named Pipes, Sockets and other IPC, Mujtaba Khambatti

Web:

http:// www.wikipedia.com
http:// www.codeproject.com
http:// www.msdn.microsoft.com
http:// www.ibm.com/
http://www.informit.com/