```
//=======================================
// Compiler: MSL
// Project:  TestIO
// Bestand:  FileCopyAsync.h
// Implemen: FileCopyAsync.cp
//=======================================

#pragma once

int main();

bool Test();

bool Test(z2 invoer_refNum,
          z4 invoer_lengte);

bool Test(z2 invoer_refNum,
          z4 invoer_lengte,
          z2 uitvoer_refNum);

bool Test(z2 invoer_refNum,
          z4 invoer_lengte,
          z2 uitvoer_refNum,
          p_char invoer);

bool Test(z2 invoer_refNum,
          z4 invoer_lengte,
          z2 uitvoer_refNum,
          p_char invoer,
          p_char uitvoer);

bool Test(z2 invoer_refNum,
          z4 invoer_lengte,
          z2 uitvoer_refNum,
          p_char invoer,
          p_char uitvoer,
          IOCompletionUPP leesCompletion);

pascal void HetIsGelezen(p_ParamBlockRec paramBlock);

pascal void HetIsGeschreven(p_ParamBlockRec paramBlock);

bool Test(z2 invoer_refNum,
          z4 invoer_lengte,
          z2 uitvoer_refNum,
```

```
          p_char invoer,
          p_char uitvoer,
          IOCompletionUPP leesCompletion,
          IOCompletionUPP schrijfCompletion);

bool VraagOmEenBlokTeLezen(z2 invoer_refNum,
                           z4 aBlok,
                           z4 overschot,
                           p_char invoer,
                           IOCompletionUPP leesCompletion);

bool VraagOmEenBlokTeSchrijven(z2 uitvoer_refNum,
                               z4 lengte,
                               p_char uitvoer,
                               IOCompletionUPP schrijfCompletion);
```

```
//=========================================
// Compiler: MSL
// Project:  TestIO
// Bestand:  FileCopyAsync.cp
//=========================================

#include "FileCopyAsync.h"

int main()
{
    cout << "Start van het programma.";
    cout << endl;

    cout << "Typ Command-Q om te stoppen.";
    cout << endl;

    c_n8 start=Tijd::Geef_microseconden();
    c_bool ok=Test();
    c_n8 einde=Tijd::Geef_microseconden();
    c_n8 duur=einde-start;
    cout << "Duur=";
    cout << duur;
    cout << endl;

    if (!ok)
        {
        cout << "Het programma mislukt.";
        cout << endl;
        }

    cout << "Einde van het programma.";
    cout << endl;

    return 0;
}

bool Test()
{
    // 1. Maak invoer spec.
    FSSpec invoerSpec;
    OSErr err=Files::OSERR_FSMakeFSSpec(0,
                                        0,
                                        "\pTemp:Invoer:PetritDemo.cw",
                                        invoerSpec);
    if (err!=osErr_noErr)
```

```
        {
        cout << "Pad van invoer is niet gevonden.";
        cout << endl;
        return false;
        }

    // 2. Open invoer.
    z2 invoer_refNum;
    err=Files::OSERR_FSpOpenDF(invoerSpec,
                               permission_rd,
                               invoer_refNum);
    if (err!=osErr_noErr)
        {
        cout << "Open data fork mislukt.";
        cout << endl;
        return false;
        }

    // 3. Zoek invoer lengte.
    z4 invoer_lengte;
    err=Files::OSERR_GetEOF(invoer_refNum,
                            invoer_lengte);
    if (err!=osErr_noErr)
        {
        cout << "Lees data fork lengte mislukt.";
        cout << endl;
        return false;
        }

    // 4. Test.
    Test(invoer_refNum,
         invoer_lengte);

    // 5. Sluit invoer.
    err=Files::OSERR_FSClose(invoer_refNum);
    if (err!=osErr_noErr)
        {
        cout << "Sluit data fork mislukt.";
        cout << endl;
        return false;
        }

    return true;
}
```

```
bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte)
{
    // 1. Maak uitvoer spec.
    FSSpec uitvoerSpec;
    OSErr err=Files::OSERR_FSMakeFSSpec(0,
                                        0,
                                        "\pKangaroo 4 GB:Test.bin",
                                        uitvoerSpec);
    if (err!=osErr_noErr)
        {
        if (err!=osErr_fnfErr)
            {
            cout << "Maak pad van uitvoer mislukt.";
            cout << endl;
            return false;
            }
        }

    // 2. Maak uitvoer.
    err=Files::OSERR_FSpCreate(uitvoerSpec,
                               'TEMP',
                               'TEMP',
                               scriptCode_system);
    if (err!=osErr_noErr)
        {
        cout << "Maak nieuw bestand mislukt.";
        cout << endl;
        return false;
        }

    // 3. Open uitvoer.
    z2 uitvoer_refNum;
    err=Files::OSERR_FSpOpenDF(uitvoerSpec,
                               permission_wr,
                               uitvoer_refNum);
    if (err!=osErr_noErr)
        {
        cout << "Open nieuwe data fork mislukt.";
        cout << endl;
        return false;
        }

    // 4. Test.
    if (invoer_lengte)
```

```
        {
        Test(invoer_refNum,
             invoer_lengte,
             uitvoer_refNum);
        }

    // 5. Sluit uitvoer.
    err=Files::OSERR_FSClose(uitvoer_refNum);
    if (err!=osErr_noErr)
        {
        cout << "Sluit nieuwe data fork mislukt.";
        cout << endl;
        return false;
        }

    return true;
}

c_z4 blok_grootte=16*1024;

bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte,
          c_z2 uitvoer_refNum)
{
    // 1. Maak invoer buffer.
    p_char invoer;
    OSErr err=Memory::OSERR_NewPtr(blok_grootte,
                                   invoer);
    if (err!=osErr_noErr)
        {
        cout << "Te weinig geheugen voor invoer.";
        cout << endl;
        return false;
        }

    // 2. Test.
    Test(invoer_refNum,
         invoer_lengte,
         uitvoer_refNum,
         invoer);

    // 3. Wis invoer buffer.
    err=Memory::OSERR_DisposePtr(invoer);
    if (err!=osErr_noErr)
        {
```

```
        cout << "Wis invoer pointer mislukt.";
        cout << endl;
        return false;
        }

    return true;
}

bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte,
          c_z2 uitvoer_refNum,
          cp_char invoer)
{
    // 1. Maak uitvoer buffer.
    p_char uitvoer;
    OSErr err=Memory::OSERR_NewPtr(blok_grootte,
                                   uitvoer);
    if (err!=osErr_noErr)
        {
        cout << "Te weinig geheugen voor uitvoer.";
        cout << endl;
        return false;
        }

    // 2. Test.
    Test(invoer_refNum,
         invoer_lengte,
         uitvoer_refNum,
         invoer,
         uitvoer);

    // 3. Wis uitvoer buffer.
    err=Memory::OSERR_DisposePtr(uitvoer);
    if (err!=osErr_noErr)
        {
        cout << "Wis uitvoer pointer mislukt.";
        cout << endl;
        return false;
        }

    return true;
}

bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte,
```

```
          c_z2 uitvoer_refNum,
          cp_char invoer,
          cp_char uitvoer)
{
    // 1. Maak UPP.
    c_IOCompletionUPP leesCompletion=NewIOCompletionUPP(HetIsGelezen);
    if (!leesCompletion)
        {
        cout << "Maak lees completion UPP mislukt.";
        cout << endl;
        return false;
        }

    // 2. Test.
    Test(invoer_refNum,
         invoer_lengte,
         uitvoer_refNum,
         invoer,
         uitvoer,
         leesCompletion);

    // 3. Wis UPP.
    DisposeIOCompletionUPP(leesCompletion);

    return true;
}

bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte,
          c_z2 uitvoer_refNum,
          cp_char invoer,
          cp_char uitvoer,
          c_IOCompletionUPP leesCompletion)
{
    // 1. Maak UPP.
    c_IOCompletionUPP schrijfCompletion=NewIOCompletionUPP(HetIsGeschreven);
    if (!schrijfCompletion)
        {
        cout << "Maak schrijf completion UPP mislukt.";
        cout << endl;
        return false;
        }

    // 2. Test.
    Test(invoer_refNum,
```

```
            invoer_lengte,
            uitvoer_refNum,
            invoer,
            uitvoer,
            leesCompletion,
            schrijfCompletion);

    // 3. Wis UPP.
    DisposeIOCompletionUPP(schrijfCompletion);

    return true;
}

v_bool isGedaan_lezen=false;
v_bool ok_lezen=false;
v_z4 aGelezen=0;

pascal void HetIsGelezen(cp_ParamBlockRec paramBlock)
{
    isGedaan_lezen=true;
    c_OSErr err_lezen=paramBlock->ioParam.ioResult;
    if (err_lezen)
        {
        ok_lezen=false;
        return;
        }
    c_z4 aGevraagd=paramBlock->ioParam.ioReqCount;
    aGelezen=paramBlock->ioParam.ioActCount;
    if (aGelezen<aGevraagd)
        {
        ok_lezen=false;
        return;
        }
    ok_lezen=true;
}

v_bool isGedaan_schrijven=false;
v_bool ok_schrijven=false;

pascal void HetIsGeschreven(cp_ParamBlockRec paramBlock)
{
    isGedaan_schrijven=true;
    c_OSErr err_schrijven=paramBlock->ioParam.ioResult;
    if (err_schrijven)
        {
```

```
        ok_schrijven=false;
        return;
        }
    c_z4 aGevraagd=paramBlock->ioParam.ioReqCount;
    c_z4 aGeschreven=paramBlock->ioParam.ioActCount;
    if (aGeschreven<aGevraagd)
        {
        ok_schrijven=false;
        return;
        }
    ok_schrijven=true;
}

bool Test(c_z2 invoer_refNum,
          c_z4 invoer_lengte,
          c_z2 uitvoer_refNum,
          p_char invoer,
          p_char uitvoer,
          c_IOCompletionUPP leesCompletion,
          c_IOCompletionUPP schrijfCompletion)
{
    z4 aBlok;
    z4 overschot;
    Blok::Verdeel(invoer_lengte,
                  aBlok,
                  overschot);

    if (!VraagOmEenBlokTeLezen(invoer_refNum,
                               aBlok,
                               overschot,
                               invoer,
                               leesCompletion))
        {
        cout << "Vraag om een blok te lezen mislukt.";
        cout << endl;
        return false;
        }

    while (aBlok or overschot)
        {
        if (isGedaan_lezen)
            {
            if (!ok_lezen)
                {
                cout << "Lezen mislukt.";
```

```
            cout << endl;
            return false;
            }
        isGedaan_lezen=false;
        if (aBlok)
            {
            aBlok--;
            }
        else
            {
            overschot=0;
            }
        cp_char r=invoer;
        invoer=uitvoer;
        uitvoer=r;
        if (!VraagOmEenBlokTeSchrijven(uitvoer_refNum,
                                       aGelezen,
                                       uitvoer,
                                       schrijfCompletion))
            {
            cout << "Vraag om een blok te schrijven mislukt.";
            cout << endl;
            return false;
            }
        if (aBlok or overschot)
            {
            if (!VraagOmEenBlokTeLezen(invoer_refNum,
                                       aBlok,
                                       overschot,
                                       invoer,
                                       leesCompletion))
                {
                cout << "Vraag om een blok te lezen mislukt.";
                cout << endl;
                return false;
                }
            }
        }

    if (isGedaan_schrijven)
        {
        if (!ok_schrijven)
            {
            cout << "Schrijven mislukt.";
            cout << endl;
```

```
                return false;
                }
            isGedaan_schrijven=false;
            }
        }

    return true;
}

bool VraagOmEenBlokTeLezen(c_z2 invoer_refNum,
                           c_z4 aBlok,
                           c_z4 overschot,
                           cp_char invoer,
                           c_IOCompletionUPP leesCompletion)
{
    static ParamBlockRec invoer_paramBlock;

    IOParam &in=invoer_paramBlock.ioParam;
    Blok::Zero(sizeof(IOParam),
               &in);
    in.ioCompletion=leesCompletion;
    in.ioRefNum=invoer_refNum;
    in.ioBuffer=invoer;
    in.ioReqCount=aBlok ? blok_grootte : overschot;
    isGedaan_lezen=false;
```

```
    c_OSErr err=Files::OSERR_PBReadAsync(invoer_paramBlock);
    if (err!=osErr_noErr)
        {
        cout << "Vraag om te lezen mislukt.";
        cout << endl;
        return false;
        }
    return true;
}

bool VraagOmEenBlokTeSchrijven(c_z2 uitvoer_refNum,
                               c_z4 lengte,
                               cp_char uitvoer,
                               c_IOCompletionUPP schrijfCompletion)
{
    static ParamBlockRec uitvoer_paramBlock;

    IOParam &uit=uitvoer_paramBlock.ioParam;
    Blok::Zero(sizeof(IOParam),
               &uit);
    uit.ioCompletion=schrijfCompletion;
    uit.ioRefNum=uitvoer_refNum;
    uit.ioBuffer=uitvoer;
    uit.ioReqCount=lengte;
    isGedaan_schrijven=false;
    c_OSErr err=Files::OSERR_PBWriteAsync(uitvoer_paramBlock);
    if (err!=osErr_noErr)
        {
        cout << "Vraag om te schrijven mislukt.";
        cout << endl;
        return false;
        }
    return true;
}
```