Manuale redatto da

http://www.giannipucillo.it
mathlink@giannipucillo.it

Manuale per *Mathlink* di *Mathematica 5.0*

Premesse

il presente manuale illustra un modo diretto per costruire l'interfacciamento verso Mathlink, basato sulle nostre esperienze. Non intendiamo estromettere i manuali, la documentazione e l'aiuto in linea forniti da Mathematica e/o dal sito della Wolfram Research. I contenuti di questo manuale sono anche *condizionati* dalle necessità dei nostri software. Non si tratta quindi di un manuale completo per *Mathlink* né, tantomeno, di un manuale approfondito o perfetto.
Nella pagina http://www.giannipucillo.it/mathlink, mettiamo a disposizione i manuali originali di Wolfram Research, gli eventuali aggiornamenti di questo manuale e una registrazione facoltativa.
Allo stato attuale, questo manuale riporta l'impiego di MathLink verso/da programmi scritti in linguaggio C.
Altre note sono riportate in calce al manuale.

Modalità di funzionamento di Mathlink

sono fondamentalmente due: la prima è quella di richiamare funzioni o programmi esterni a Mathematica, direttamente dal front-end di Mathematica: abbiamo nominato questa modalità *MathCall* (paragrafo 1). La seconda è quella di appoggiare il proprio software (a cui ci riferiremo, nel corso del manuale con *software* o *programma*) ai complessi calcoli del kernel di Mathematica, nominata *MathCalc* (paragrafo 2).

1 La modalità *MathCall*:

La modalità per chiamare funzioni da Mathematica, si divide in tre passi, come descritto nei seguenti sotto-paragrafi. Di seguito riportiamo la funzione loop principale di funzionamento:

```
int WINAPI WinMain(HINSTANCE hInst,HINSTANCE hPrevInst,LPSTR lpCmdLine,int nCmdShow)
{
    MSG Msg;
    link=(MLINK)0;
    env=(MLENV)0;

    if (IdleInit(hInst) != NULL) //Inizializzazione icona nella barra delle applicazioni
    {
        /* Definizione collegamento a Mathlink */
        if (MathLink50Open() == FALSE)
        {
            /* Se il collegamento è già stato attivato da Mathematica,la funzione fallisce */
            MathLink50Close();
            return(0); //uscita dal programma
        }

        /* Loop continuo fino alla chiusura manuale del programma,oppure fino alla esecuzione
            della funzione Install di Mathematica. */
        while (MathLink50Connect() == FALSE)
        {
            if(PeekMessage(&Msg,(HWND)0,0,0,PM_REMOVE))
            {
                TranslateMessage(&Msg);
                DispatchMessage(&Msg);
                if (Msg.message == WM_QUIT)
                {
                    MathLink50Close();
                    return(0); //uscita dal programma
                }
            }
        }

        /* Fino a quando il link è valido,loop per ricevere le chiamate da Mathematica */
        while (link > 0)
```

```
        {
            MathLink50ReceiveData(); //Riceve le chiamate da Mathematica
            if(PeekMessage(&Msg,(HWND)0,0,0,PM_REMOVE))
            {
                TranslateMessage(&Msg);
                DispatchMessage(&Msg);
                if (Msg.message == WM_QUIT)
                {
                    MathLink50Close();
                    return(0); //uscita dal programma
                }
            }
        }
    }
    return(0); //uscita dal programma
}
```

La struttura del programma è predisposta a non invalidare il loop principale che deve essere utilizzato per il sistema operativo ospitante. In questo modo e con la funzione successiva *MathLink50Open()*, se il kernel di Mathematica viene abbattuto, anche il programma verrà concluso. Il kernel di Mathematica tramite il front-end, può comunque indicare lo scollegamento dal programma con l'istruzione:

```
Uninstall[vlink];
```

Anche in questo caso, il programma si conclude.

1.1 stabilire la connessione con il programma

```
BOOL MathLink50Open(sTypeInConnection *psInConnection)
{
    char *argv[10];

    env = MLInitialize(0);
    if (env == 0)
    {
        //mcSendStringToMsgWarningWin("Inizializzazione fallita per MathLink 5.0.");
        MathLink50Close();
        return(FALSE);
    }

    /* Il primo argomento deve essere "-linkname" */
    strcpy(argv1,"-linkname");

    /* Il nome della connessione deve essere sempre definito in formato stringa */
#if defined(TCP) || defined(TCPIP)
    strcpy(argv2,"6000@Master");
#else
    /* In questo caso è necessario specificare anche il nome del computer: "Master" */
    strcpy(argv2,"6000");
#endif

    /* Il terzo argomento deve essere "-linkprotocol"*/
    strcpy(argv3,"-linkprotocol");
#if defined(SHAREDMEM)
    strcpy(argv4,"SharedMemory");
#elif defined(TCPIP)
    strcpy(argv4,"TCPIP");
#elif defined(FILEMAP)
    strcpy(argv4,"FileMap");
#else
```

```
    strcpy(argv4,"TCP");
#endif

    /* Il quinto argomento deve essere "-linkmode" */
    strcpy(argv5,"-linkmode");

    /* Il sesto argomento lo abbiamo fissato a "-listen" */
    strcpy(argv6,"listen");

    argv[0]=argv1;
    argv[1]=argv2;
    argv[2]=argv3;
    argv[3]=argv4;
    argv[4]=argv5;
    argv[5]=argv6;
    link = MLOpenArgv(env,argv,argv+6,&ErrNum);
    if (link == 0)
    {
        //mcSendStringToMsgWarningWin("Opening connection failed for MathLink 5.0.");
        MathLink50Close();
        return(FALSE);
    }
    MLFlush(link);

    return(TRUE);
}
```

La funzione pone in attesa il programma fino al collegamento da parte di Mathematica tramite il Mathlink. Questo modo permette al software di creare il collegamento senza fermarsi in attesa di *acknoledge* da parte di Mathematica. Il nome della connessione “6000” e il protocollo di trasmissione, devono coincidere tra il software e l’istruzione al collegamento di Mathematica. Se il collegamento avviene tramite TCP o TCP/IP, aggiungere al nome della connessione il simbolo “@” seguito dal nome del computer su cui Mathematica è in funzione.
La funzione eseguita dal front-end di Mathematica è

```
vlink=LinkOpen["6000",LinkProtocol®"SharedMemory",LinkMode®Connect];
```

Quando Mathematica avrà stabilito la connessione, sarà necessario eseguire la seguente funzione che completa la connessione installando le funzioni del programma all’interno di Mathematica:

```
BOOL MathLink50Connect(void)
{
    int iM;

    if (MLReady(link) != 0)
    {
        if (MLConnect(link) == 0)
        {
            MathLink50Close();
            return(FALSE);
        }
    }
    else
    {
        return(FALSE);
    }

    iM=MLPutFunction(link,"DefineExternal",(long)3);
    iM=MLPutString(link,"Fn[Data_List]");
    iM=MLPutString(link,"{Data}");
    iM=MLPutInteger(link,0);
    iM=MLError(link);
    if (iM != 0)
```

```
    {
        //mcSendStringToMsgInfoWin("Error from MLConnect: %s",(UCHAR *)MLErrorMessage(link));
        MLClearError(link);
        MLNewPacket(link);
        return(FALSE);
    }

    iM=MLPutSymbol(link,"End");
    iM=MLFlush(link);
    return(TRUE);
}
```

Mathematica segnalerà alla funzione *MLReady()* che è pronto per effettuare l'installazione delle funzioni del programma, con la seguente istruzione eseguita da front-end:

```
Install[vlink];
```

La funzione *MathLink50Connect()* installa una sola funzione che in questo caso è stata chiamata *Fn*. Per verificare che la funzione sia presente in Mathematica, va eseguita la funzione

```
In[3]= LinkPatterns[vlink]

Out[3]= {Fn[Data_List]}
```

senza punto e virgola, che riporterà la lista di funzioni che sono state installate.

1.2 chiamare le funzioni del programma

La chiamata continua alla funzione *MathLink50ReceiveData()* dal loop principale del programma, è quella incaricata a ricevere e ordinare i dati in arrivo da Mathematica:

```
void MathLink50ReceiveData(void)
{
    float fFloat=0.0f;
    static char chFlag=FALSE;
    int iM,i=0,iGetNext,iQty=0;
    long lData=0;

    MLFlush(link);
    if (MLReady(link) != 0)
    {
        if ((iM=MLNextPacket(link)) == CALLPKT)
        {
            if (MLGetInteger(link,&i) == 0)
                return;
            if (MLCheckFunction(link,"List",&lData) == 0)
                return;
            if (i > 0)
            {
                MLClearError(link);
                MLPutSymbol(link,"$Failed");
                MLEndPacket(link);
                MLNewPacket(link);
                return;
            }
            if ((iM=MLReady(link)) != 0)
            {
                iGetNext=MLGetNext(link);
                switch (iGetNext)
                {
                    case MLTKERR:
                        iM=MLError(link);
                        if (iM != 0)
                        {
```

```
                    //mcSendStringToMsgInfoWin("Error from Mathematica: %s",
                                                    (UCHAR *)MLErrorMessage(link));
                    MLClearError(link);
                    MLNewPacket(link);
                }
            break;

            case MLTKREAL:
            iM=MLGetFloat(link,&fFloat);
            break;

            case MLTKSTR:
            iM=MLGetString(link,&ppch);
            if (iM != 0)
            {
                //mcSendStringToMsgInfoWin(ppch);
                MLDisownString(link,ppch);
                return;
            }
            break;

            case MLTKINT:
            iM=MLGetInteger(link,&i);
            break;

            case MLTKFUNC:
            iM=MLGetFunction(link,&ppch,&longMathLinkQtyArgToFunction);
            if (iM != 0)
                GetFunction(longMathLinkQtyArgToFunction);
            break;

            case MLTKSYM:
            iM=MLGetSymbol(link,&ppch);
            if (iM != 0)
            {
                //mcSendStringToMsgInfoWin(ppch);
                MLDisownSymbol(link,ppch);
            }
            break;

            default:
                MLNewPacket(link);
            break;
        }
        MLNewPacket(link);
        MLFlush(link);
    }
    iGetNext=1;
    iM=MLNewPacket(link);
    iM=MLPutInteger(link,iGetNext);
    iM=MLEndPacket(link);
    iM=MLNewPacket(link);
}
else
{
    iGetNext=MLGetNext(link);
    switch (iGetNext)
    {
        case MLTKERR:
            iM=MLError(link);
            if (iM != 0)
```

```
                {
                    //mcSendStringToMsgInfoWin("Error from Mathematica: %s",
                                                (UCHAR *)MLErrorMessage(link));
                    MLClearError(link);
                    MLNewPacket(link);
                    if (iM == MLECLOSED || iM == MLEDEAD)
                    {
                        MathLink50Close();
                        return;
                    }
                }
                break;
            }
        }
    }
}
```

MathLink50ReceiveData() si avvale della funzione recursiva *GetFunction()*:

```
void GetFunction(long lPassedMathLinkQtyArgToFunction)
{
    int iM,i=0,iGetNext,iQty=0;
    long lData=0,lLocalMathLinkQtyArgToFunction;

    lPassedMathLinkQtyArgToFunction;
    MLDisownSymbol(link,ppch);
    for (lData=0;lData < lPassedMathLinkQtyArgToFunction;lData++)
    {
        iGetNext=MLGetNext(link);
        switch (iGetNext)
        {
            case MLTKREAL:
            iM=MLGetFloat(link,(float *)pCore);
            break;

            case MLTKSTR:
            iM=MLGetString(link,&ppch);
            if (iM != 0)
            {
                iM=strlen(ppch);
                if (iM > 0)
                {
                    if (iM < 256)
                        memcpy(pCore,ppch,iM);
                    else
                        memcpy(pCore,ppch,256);
                }
                MLDisownString(link,ppch);
            }
            break;

            case MLTKINT:
            iM=MLGetInteger(link,&i);
            break;

            case MLTKFUNC:
            iM=MLGetFunction(link,&ppch,&lLocalMathLinkQtyArgToFunction);
            if (iM != 0)
                GetFunction(lLocalMathLinkQtyArgToFunction);
            break;
        }
```

```
    }
}
```

La funzione *Fn* installata, è una funzione che abbiamo voluto fosse generalizzata, ovvero è una funzione che accetta una lista di valori come parametri passati e ritorna una lista di valori al chiamante. Le liste possono essere composte da un qualsiasi numero di elementi, anche nessuno, e di qualsiasi tipo. La tipologia di funzione è stata definita nella funzione *MathLink50Connect()* a partire dall'istruzione *MLPutFunction(link,"DefineExternal",(long)3);* e conclusasi con l'istruzione *MLPutSymbol(link,"End");*.

Esempio 1: la funzione viene chiamata dal front-end di Mathematica per passargli quattro parametri: un *floating-point*, due *interi*, e una *stringa*.

```
Fn[{1.2,10,-2000,"stringa"}]
```

Esempio 2: la funzione viene chiamata ma non gli viene passato alcun parametro.

```
Fn[{}]
```

Esempio 3: la funzione viene chiamata con un *codice intero* passato come argomento; questa volta è attesa una lista di valori, anche di tipo mescolato.

```
A={};A=Fn[{132}]
```

In un contesto dove le funzioni da chiamare sono diverse, il concetto del codice è stato da noi adottato per indirizzare le funzioni: è sufficiente costruire un vettore di funzioni oppure utilizzare una istruzione condizionale.
Il passaggio dei parametri e la relativa discriminazione della funzione da eseguire si risolve quindi abbastanza semplicemente. Il punto migliore ove rilevare il codice della funzione da eseguire è nel blocco seguente della funzione *GetFunction()*:

```
        case MLTKINT:
        iM=MLGetInteger(link,&i);
        FnCode=i;
        break;
```

La funzione *GetFunction()* è incaricata di rilevare, smistare e memorizzare tutti i parametri passati alla funzione *Fn* che poi dovrà eseguire la funzione voluta. Fare quindi attenzione a due punti:

- il primo parametro deve essere sempre un intero
- il primo parametro deve essere sempre l'identificativo della funzione
- gli altri parametri potrebbero essere degli interi, rischiando di sovrapporre il codice della funzione

Per quanto concerne la restituzione a Mathematica dei valori calcolati dalla funzione del programma, il blocco seguente della funzione *MathLink50ReceiveData()* illustra brevemente come deve essere effettuato:

```
        iGetNext=1;
        iM=MLNewPacket(link);
        iM=MLPutInteger(link,iGetNext);
        iM=MLEndPacket(link);
        iM=MLNewPacket(link);
```

In questo caso viene restituito un valore intero precedentemente memorizzato nella variabile *iGetNext*; per restituire più valori dello stesso tipo, conviene utilizzare il seguente esempio:

```
    MLNewPacket(link);
    MLPutFunction(link,"List",3);
    MLPutFloat(link,10.9);
    MLPutFloat(link,11.22);
    MLPutFloat(link,fFloat);
    MLEndPacket(link);
    MLNewPacket(link);
```

```
MLFlush(link);
```

La funzione *MLPutFunction(link,"List",3);* istruisce Mathematica ad aspettarsi una lista con tre valori. Quando i valori da inviare sono di tipo diverso, fare riferimento al seguente esempio:

```
MLNewPacket(link);
MLPutFunction(link,"List",3);
MLPutInteger(link,10);
MLPutFloat(link,11.22);
MLPutString(link,”string short”);
MLEndPacket(link);
MLNewPacket(link);
MLFlush(link);
```

1.3 chiudere la comunicazione con il programma

Riportiamo di seguito la funzione di chiusura della comunicazione tra il programma e Mathematica:

```
BOOL MathLink50Close(void)
{
    if(link != 0)
    {
        MLClose(link);
        link=0;
    }
    if(env != 0)
    {
        MLDeinitialize(env);
        env=0;
    }
}
```

2 La modalità *MathCalc*:

(Questa modalità è attualmente in corso di stesura.)

Note

- I programmi *MathCall.c* e *MathCalc.c* completi possono essere scaricati dal nostro sito dalla pagina http://www.giannipucillo.it/mathlink, previa compilazione del modulo che ci fornirà il vostro indirizzo e-mail (ed eventuali suggerimenti, richieste, note, etc.). Il vostro indirizzo di posta verrà utilizzato solo dalla ditta Gianni Pucillo e solo per informarvi sui nostri prodotti e, in particolar modo, sulle evoluzioni e sugli aggiornamenti a questo manuale: nel completo rispetto della legge sulla privacy.
- Ricordiamo che il presente manuale e i sorgenti messi a disposizione sul sito, oltre al divieto di utilizzo a scopi di lucro, non sono vincolati in nessuna maniera e sono liberi di essere modificati, copiati, distribuiti.
- Questo manuale è stato prodotto su richiesta. Potete inoltrarci domande o richieste: cercheremo sicuramente di rispondervi anche per aumentare la quantità di informazioni (fino ad ora molto ristrette).
- I principali prodotti che utilizzano un insieme più articolato di interfacciamento, sono *VisProject*, *PCAnalyze*^{N} e, in particolar modo, l'interprete OpenGL per Mathematica: *MathInt*.
- *Sia il software di interfacciamento che eventuali particolari esigenze, possono essere valutate dal nostro staff per lo sviluppo su commissione.*

Lo staff della *Gianni Pucillo* vi augura Buon lavoro.