Novell Developer Kit

www.novell.com

SAMPLE CODE

March 1, 2006

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

Third-Party Materials

Contents

About This Guide

This guide contains the following sections:

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Documentation Updates

For the most recent version of this guide, see NLM and NetWare Libraries for C (including CLIB and XPlat) (http://developer.novell.com/ndk/clib.htm)

Additional Information

For information about other CLib and XPlat interfaces, see the following guides:

- *NDK: NLM Development Concepts, Tools, and Functions*
- *NDK: Program Management*
- *NDK: NLM Threads Management*
- *NDK: Connection, Message, and NCP Extensions*
- *NDK: Multiple and Inter-File Services*
- *NDK: Single and Intra-File Services*
- *NDK: Volume Management*
- *NDK: Client Management*
- *NDK: Network Management*
- *NDK: Server Management*
- *NDK: Unicode*
- *NDK: Getting Started with NetWare Cross-Platform Libraries for C*
- *NDK: Bindery Management*
- *NDK: Internationalization*

For CLib source code projects, visit Forge (http://forge.novell.com).

For help with CLib and XPlat problems or questions, visit the NLM and NetWare Libraries for C (including CLIB and XPlat) Developer Support Forums (http://developer.novell.com/ndk/devforums.htm). There are two for NLM development (XPlat and CLib) and one for Windows XPlat development.

Documentation Conventions

In this documentation, a greater-than symbol (>) is used to separate actions within a step and items within a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Example Code 1

This book contains sample code for using CLib functions and XPlat functions. See

- Section 1.1, "CLib Examples," on page 11
- Section 1.2, "XPlat Examples," on page 60

For additional samples, see NLM and NetWare Libraries for C Sample Code (http://developer.novell.com/ndk/doc/samplecode/clib_sample/index.htm).

1.1 CLib Examples

The following examples illustrate the CLib functions. For additional examples, see NLM and NetWare Libraries for C (http://developer.novell.com/ndk/clib_sample.htm).

1.1.1 Adding to OS Supported Language List: Example

```
NOTE: taken from ADDLANG.C in the \EXAMPLES directory


/*********************************************************************
  ADDLANG.C
**********************************************************************

  This example demonstrates how to add a language to the OS supported
  language list.

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/string.h>
#include <nlm/nwthread.h>
#include <nlm/nwadv.h>

#define MAX_LANGUAGE_ID      999
#define MIN_NON_NETWARE_ID   100
#define MAX_LANGUAGE_NAME    255

int main(void)
{
   int     ID = MIN_NON_NETWARE_ID;
   int     len;
   int     ccode = TRUE;
   char    **messageTable;
   LONG    messageCount = 0;
   LONG    languageID;
   BYTE    languageName[MAX_LANGUAGE_NAME];
   BYTE    name[MAX_LANGUAGE_NAME];
```

```
/* Multi-Language enabled NLM applications can load tables of
   messages for different languages by using this API. Default
   Language tables can be bound to NLM applications with the
   messages option in NLMLINK or NLMLINKP.  */

LoadLanguageMessageTable(&messageTable,
                         &messageCount,
                         &languageID);

/* NLMs can then access messages through the table like this...*/

if(messageCount)
   printf("The first message in our table is %s.\n",
           messageTable[1]);

languageID = GetCurrentOSLanguageID();
if(!ReturnLanguageName(languageID, languageName))
{
   printf("The current language on this server is %i : %s\n",
          languageID, languageName);
}

/*...find the next available ID number...*/
while(ccode && ID < MAX_LANGUAGE_ID)
{
   ccode = AddLanguage(ID, "TEST LANGUAGE", TRUE);
   if (ccode) ID++;
}
if (!ccode)
{
   printf("Test language %i added to OS language list.\n", ID);
   strcpy(name, "PIG LATIN 0");

   /*...find a unique name for this language...*/
   ccode = TRUE;
   while(ccode)
   {
      ccode = RenameLanguage(ID, name, TRUE);
      if (ccode)
      {
         len = strlen(name);
         name[len - 1]++;
      }
   }
   if(!ccode)
   {
      ReturnLanguageName(ID, languageName);
     printf("The name of language %i is %s.\n", ID, languageName);
      SetCurrentOSLanguageID(ID);
      languageID = GetCurrentOSLanguageID();

      if(!ReturnLanguageName(languageID, languageName))
        printf("The current language on this server is %i : %s\n",
                    languageID, languageName);
      }
      else
         printf("Unable to rename test language.\n");
   }
   else
      printf("Unable to add test language to OS language list.\n");

   return(0);
}
```

1.1.2 Calendar program: Example

NOTE: taken from CALENDAR.C in the \EXAMPLES directory

```
/**********************************************************************
   CALENDAR.C
**********************************************************************

  This program displays three calendars on the screen; the current
  month, the previous month, and the next month.

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/nwtypes.h>
#include <nlm/nwconio.h>
#include <nlm/string.h>
#include <nlm/time.h>

#define FEBRUARY    1
#define NARROW      3
#define WIDE        4

#define PosCursor( row, col )   gotoxy(col,row)
#define ClearScreen()           clrscr()

static int   Jump[ 12 ]      =   { 1, 4, 4, 0, 2, 5, 0, 3, 6, 1, 4, 6 };
static int   MonthDays[ 12 ] =   { 31, 28, 31, 30, 31, 30, 31, 31, 30,
                                    31, 30, 31 };
static char *MonthName[ 12 ] =   { "January", "February", "March",
                                    "April", "May", "June", "July",
                                    "August", "September", "October",
                                    "November", "December" };
static char  *WideTitle      =   { "Sun Mon Tue Wed Thu Fri Sat" };
static char  *NarrowTitle    =   { "Su Mo Tu We Th Fr Sa" };


int main(void)
{
   time_t               curr_time;
   register struct tm   *tyme;

   ClearScreen();

   /* get today’s date */

   curr_time = time( NULL );
   tyme = localtime( &curr_time );

   /* draw calendar for this month */

   Calendar( tyme->tm_mon, tyme->tm_year, 10, 26, WIDE, WideTitle );

   /* draw calendar for last month */

   tyme->tm_mon–;
   if( tyme->tm_mon < 0 )
   {
      tyme->tm_mon = 11;
      tyme->tm_year–;
   }

   Calendar( tyme->tm_mon, tyme->tm_year, 5, 3, NARROW, NarrowTitle );

    /* draw calendar for next month */

   tyme->tm_mon += 2;
   if( tyme->tm_mon > 11 )
   {
      tyme->tm_mon -= 12;
      tyme->tm_year++;
   }

   Calendar( tyme->tm_mon, tyme->tm_year, 5, 56, NARROW, NarrowTitle );

   PosCursor( 20, 1 );
   return(0);
}

void Calendar(int month, int year, int row, int col,
              int width, char *title)
{
   register int    start;
   register int    days;
   register int    box_width;
   register char   *str;
   register int    i;

   box_width = 7 * width - 1;
   Box( row, col, box_width, 8 );
   str = MonthName[ month ];
   PosCursor( row - 1, col + 1 + ( box_width - strlen( str ) - 5 ) / 2
);
```

```
   printf( "%s 19%d\n", str, year );
   fflush( stdout );
   PosCursor( row + 1, col + 1 );
   printf( title );
   fflush( stdout );

   start = year + year / 4 + Jump[ month ];

   if( ( year % 4 == 0 ) && ( month <= FEBRUARY ) )
   {
      –start;
   }
   start = start % 7 + 1;
   if( ( year % 4 == 0 ) && ( month == FEBRUARY ) )
   {
      days = 29;
   }
   else
   {
      days = MonthDays[ month ];
   }
   row += 3;
   for( i = 1; i <= days; ++i )
   {
      PosCursor( row, col + width * start - 2 );
      printf( "%2d", i );
      fflush( stdout );
      if( start == 7 )
      {
         printf( "\n" );
         fflush( stdout );
         ++row;
         start = 1;
      }
      else
      {
         ++start;
      }
   }
}

void Box(int row, int col, int width, int height )
{
   register int   i;

   Line( row, col, width, ’Ú’, ’Ä’, ’¿’ );
   Line( row + 1, col, width, ’’, ’ ’, ’’ );
   Line( row + 2, col, width, ’Ã’, ’Ä’, ’´’ );

   for( i = 3; i <= height; ++i )
   {
      Line( row + i, col, width, ’’, ’ ’, ’’ );
   }
   Line( row + height + 1, col, width, 'À', 'Ä', 'Ù' );
}

void Line( int row, int col, int width,
           char left, char centre, char right )
{
   char   buffer[ 80 ];

   buffer[ 0 ] = left;
   memset( &buffer[ 1 ], centre, width );
   buffer[ width + 1 ] = right;
   buffer[ width + 2 ] = '\0';
   PosCursor( row, col );
   printf( buffer );
   fflush( stdout );
}
```

1.1.3 Adding Console Commands: Example

NOTE: taken from CONCOM.C in the \EXAMPLES directory

```
/**********************************************************************
   CONCOM.C
***********************************************************************

   This module shows how to add console commands to NetWare.  This is
   useful in situations where an NLM needs console IO support
   (for instance, an NLM which has no screen).

   After loading CONCOM.NLM, return to the system console and notice
   that two new commands are accepted by the NetWare console - CONCOM
   STAT and CONCOM MISC.

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/string.h>
#include <nlm/nwtypes.h>
#include <nlm/nwthread.h>
#include <nlm/nwerrno.h>
#include <nlm/nwadv.h>
#include <nlm/nwconio.h>

#define HANDLEDCOMMAND  0
#define NOTMYCOMMAND    1

/* Prototypes */

int     InstallConsoleHandler(void);
```

```
static  void  CCRemoveConsoleHandler(void);
void    HandleStatRequest(void *dummy);
void    HandleMiscRequest(void *dummy);
void    ScheduleStatCommand(void);
void    ScheduleMiscCommand(void);
static LONG CommandLineInterpreter(LONG screenID, BYTE *commandLine );

/* Globals */

static  LONG    StatCommandThread;
static  LONG    MiscCommandThread;

/*  Structure used to register/deregister a console handler with the OS
*/
static  struct  commandParserStructure ConsoleHandler =
                                     {0, CommandLineInterpreter, 0 };

int main(void)
{
   if( BeginThread(HandleStatRequest,NULL,8192,NULL) == EFAILURE )
   {
      printf("ConCom: Couldn’t start Handle Stat Request thread!\n");
      exit(1);
   }

   if( BeginThread(HandleMiscRequest,NULL,8192,NULL) == EFAILURE )
   {
      printf("ConCom: Couldn’t start Handle Misc Request thread!\n");
      exit(1);
   }

   ThreadSwitch();     /* let them run at least once... */

   if( InstallConsoleHandler() == EFAILURE )
   {
      printf("ConCom: Couldn’t install command parser!\n");
      exit( 1 );
   }

   ExitThread(EXIT_THREAD,0);
   return(0);
}

static LONG CommandLineInterpreter(LONG screenID, BYTE *commandLine )
{
   /*  All added commands begin with "CONCOM " */

   if( !strnicmp("CONCOM ",commandLine,7) )
   {
      /*
        Figure out which command it is, and then schedule the
        appropriate thread to handle the request.  It,s a good
        idea to execute quickly here, and return back to the
        console handler, so all I’m doing is signalling a local
```

```
          semaphore which will wake up the appropriate function.

          If you want to do all of the work here, you MUST change
          the CLib context to that of a thread group in your NLM.
          You can use SetThreadGroupID to do this.  Remember to
          switch it back before you return.
       */

       if( !strnicmp("STAT",&commandLine[7],4) )
       {
          ScheduleStatCommand();
          ConsolePrintf("ConCom: STAT request being processed!\r\n");
       }
       else if( !strnicmp("MISC",&commandLine[7],4) )
       {
          ScheduleMiscCommand();
          ConsolePrintf("ConCom: MISC request being processed!\r\n");
       }
       else
       {
          ConsolePrintf("ConCom: \"%s\" is not a valid command!\r\n",
                         &commandLine[7]);
       }

       /*  Tell NetWare we handled the command */
       return HANDLEDCOMMAND;
    }

    /*  Tell NetWare that the command isn't mine */
    return NOTMYCOMMAND;
}

/*  This function is called during NLM shutdown */

static void CCRemoveConsoleHandler(void)
{
    UnRegisterConsoleCommand( &ConsoleHandler );
}

/*  This function installs the handler */

int InstallConsoleHandler(void)
{
    /*  Our command line handler interfaces the system operator
        with this NLM */

    ConsoleHandler.RTag = AllocateResourceTag(GetNLMHandle(),
                                             "Command Line Processor",
                                             ConsoleCommandSignature);
    if( ! ConsoleHandler.RTag)
    {
       printf("Error on allocate resource tag\n");
       return EFAILURE;
    }
```

```
   RegisterConsoleCommand( &ConsoleHandler );

   /*  The Remove procedure unregisters the console handler */

   atexit( CCRemoveConsoleHandler );
   return ESUCCESS;
}

void ScheduleStatCommand(void)
{
   ResumeThread(StatCommandThread);
}

void HandleStatRequest(void *dummy)
{
   for(;;)
   {
      SuspendThread(StatCommandThread = GetThreadID() );
      ConsolePrintf("ConCom: Message from the STAT command\n");
   }
}

void ScheduleMiscCommand(void)
{
   ResumeThread(MiscCommandThread);
}

void HandleMiscRequest(void *dummy)
{
   for(;;)
   {
      SuspendThread(MiscCommandThread = GetThreadID() );
      ConsolePrintf("ConCom: Message from the MISC command\n");
   }
}
```

1.1.4 Using DOWN_SERVER Event: Example

```
NOTE: taken from DOWNHAND.C in the \EXAMPLES directory


/**********************************************************************
   DOWNHAND.C
**********************************************************************

  DOWNHAND.C is an example NLM that demonstrates using the DOWN_SERVER
  event to warn the operator before allowing the server to go down.
  Enter 0 at the command line to allow the server to go down.  Enter 1
  to prompt the operator before allowing the server to go down.
  Usage: DOWNHAND <arg>
          0  Allow the server to go down
          1  Prompt operator before downing server

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/nwthread.h>
#include <nlm/nwadv.h>

/* Prototypes */
LONG WarnProcedure(void (*OutPutFunc)(void *fmt,...),LONG parameter);
void ReportProcedure(LONG parameter);
void ExitProcedure(void);

/* Globals */
int    RetValueFromWarnProc=0;     /* Let event continue by default */
LONG   eventHandle;                /* The DOWN_SERVER event handle */


int main(int argc, char *argv[])
{
   if(argc > 1)
      RetValueFromWarnProc = atoi(argv[1]);

   atexit(ExitProcedure);      /* register atexit procedure */

   eventHandle = RegisterForEvent(EVENT_DOWN_SERVER,
                                  ReportProcedure,
                                  WarnProcedure);
   if(eventHandle == -1)
   {
      printf("Error registering event DOWN_SERVER!\n");
      return(1);
   }

   printf("Event registration is complete. Attempt DOWN now...\n");
   ExitThread(TSR_THREAD,0);
   return(0);
}

/*
  WarnProcedure() gets called before the event DOWN occurs. This
  gives you a chance to have the OS warn the operator before allowing
  the event to continue.  In order to get NetWare to display the
  warning message, return a non-zero value.
*/

LONG WarnProcedure(void (*OutPutFunc)(void *fmt,...),LONG parameter)
{
   OutPutFunc("Inside WarnProcedure()\r\n");
   OutPutFunc("Returning %u to NetWare\r\n",RetValueFromWarnProc);
   return RetValueFromWarnProc;
}

/*
  ReportProcedure() gets called after the event DOWN occurs. This
  basically is a signal to your NLM that the server has been downed.
  The parameter for this event is undefined.
*/

void ReportProcedure(LONG parameter)
{
   printf("Inside ReportProcedure()\r\nThe server has been
           downed\r\n");
}

/*
  This procedure is called when the NLM is unloaded or exits.
*/

void ExitProcedure(void)
{
   printf("Unloading DOWNHAND NLM.\r\n");

   if(eventHandle != NULL && UnregisterForEvent(eventHandle) != NULL)
      printf("Error during Unregister of event DOWN_SERVER\r\n");
}
```

1.1.5 Using the MODULE_UNLOAD Event: Example

```
NOTE: taken from NOUNLOAD.C in the \EXAMPLES directory


/**********************************************************************
   NOUNLOAD.C
***********************************************************************

  NOUNLOAD.C is an example NLM that demonstrates the use of the
  MODULE_UNLOAD event.  This event can be used to detect when an NLM is
  being unloaded.  Enter 0 at the command line to allow the NLM to be
  unloaded.  Enter 1 to prompt the operator before unloading the NLM.

  Usage: NoUnload <arg>
            0   Allow the NLM to be unloaded
            1   Prompt operator before unloading

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/nwthread.h>
#include <nlm/nwadv.h>
```

```
/* Prototypes */
LONG   WarnProcedure(void (*OutPutProc)(void *fmt, ...),
                     LONG UnloadID);
void   ReportProcedure(LONG UnloadID);
void   UnloadProcedure(void);

/* Globals */
LONG   eventHandle;           /* Handle for the MODULE_UNLOAD event */
LONG   RetValueFromWarnProc;  /* What’s returned from the WarnProc  */
LONG   myNLMID;               /* This is the ID of my NLM           */

int main(int argc, char *argv[])
{
   if(argc > 1)
      RetValueFromWarnProc = atoi(argv[1]);

   atexit(UnloadProcedure);

   eventHandle = RegisterForEvent(EVENT_MODULE_UNLOAD,
                                  ReportProcedure,
                                  WarnProcedure);
   if(eventHandle == -1)
   {
      printf("Failure to Register for event MODULE_UNLOAD!\n");
      exit(1);
   }
   /*
     Get the NLM ID of this NLM.  Use this to determine what
     value is returned from the WarnProcedure.  i.e. I only want
     to have control over the unloading of this NLM.
   */
   myNLMID = FindNLMHandle("NOUNLOAD");
   printf("Go ahead and attempt unload...\n");
   ExitThread(TSR_THREAD,0);       /* TSR this NLM. */

   return(0);
}

/*
  This procedure is called before the module is unloaded.  A non-zero
  return value will cause NetWare to prompt the operator before
  unloading the module.
*/

LONG WarnProcedure(void (*OutPutProc)(void *fmt, ...), LONG UnloadID)
{
   OutPutProc("Inside WarnProcedure\r\n");
   if(UnloadID == myNLMID)
   {
      OutPutProc("Parameter is %x\r\n",UnloadID);
      return RetValueFromWarnProc;
   }
   return 0;   /* Don’t care about other NLMs getting unloaded */
}
```

```
/*
  The ReportProcedure is called after the NLM was unloaded.
*/

void ReportProcedure(LONG UnloadID)
{
   printf("Inside ReportProcedure\r\nNLMID is %x\r\n",UnloadID);
}

/*
  This is a generic unload procedure.  It simply unregisters the
  event handling routines we previously registered.
*/

void UnloadProcedure(void)
{
   if(UnregisterForEvent(eventHandle) != NULL)
      printf("Failure to unregister MODULE_UNLOAD event!\n");
}
```

1.1.6 Using the TRUSTEE_CHANGE Event: Example

```
NOTE: taken from TRUSTEE.C in the \EXAMPLES directory


/**********************************************************************
   TRUSTEE.C
**********************************************************************

  This module illustrates the use of the TRUSTEE_CHANGE event.  This
  event is raised when a trustee is added or deleted to/from the file
  system.

  NOTE:  This NLM REQUIRES NetWare 3.11 and CLib 3.11 or later.

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/obsolete/fileengd.h>
#include <nlm/nwdir.h>
#include <nlm/nwsemaph.h>
#include <nlm/nwerrno.h>
#include <nlm/nwadv.h>

/* Prototypes */
void ReportProcedure(LONG parameter);
void UnloadProcedure(void);
void NetWarePathToDOSPath(BYTE *psp, LONG pathCount);
```

```
/* Globals */
LONG    eventHandle;       /* The TRUSTEE_CHANGE event handle     */
LONG    semaphoreHandle;   /* Main thread blocked on this waiting */

/*
  Save the data passed to the report routine in this structure.
  Then signal a semaphore to wake up the main thread, it will print
  the data in the structure.
*/

struct EventTrusteeChangeStruct etc;

int main(void)
{
   BYTE pathString[255];
   BYTE volumeName[32];
   LONG pathCount;

   semaphoreHandle = OpenLocalSemaphore(0);
   if(semaphoreHandle == -1 )
   {
      printf("Could not open a local semaphore\n");
      return(1);
   }

   atexit(UnloadProcedure);      /* register unload procedure */

   eventHandle = RegisterForEvent(EVENT_TRUSTEE_CHANGE,
                                  ReportProcedure,
                                  NULL);
   if(eventHandle == -1)
   {
      printf("Error registering event TRUSTEE_CHANGE!\n");
      return(1);
   }
   printf("Event registration is complete.\r\n");

   /*  Main Loop. Wait for work. */

   while(1)
   {
      WaitOnLocalSemaphore( semaphoreHandle );

      /*  A Trustee has been added or deleted, print the info */

     printf("Trustee change detected for object %x\r\n",etc.objectID);
      switch( etc.changeFlags )
      {
         case EVENT_NEW_TRUSTEE:
            printf("Trustee added to file system. Rights granted
                    %x\r\n", etc.newRights);
            break;

         case EVENT_REMOVE_TRUSTEE:
            printf("Trustee removed from file system.\r\n");
            break;

         default:
            printf("Unknown changeFlags %x.\r\n",etc.changeFlags);
            break;
      }
      if(NULL == FEMapVolumeAndDirectoryToPath(etc.volumeNumber,
                                               etc.entryID,
                                               pathString,
                                               &pathCount))
      {
         /*  Convert the NetWare-Style path to a DOS-Style path */

         NetWarePathToDOSPath( pathString, pathCount );
         GetVolumeName(etc.volumeNumber,volumeName);
         printf("Change made to %s%s\r\n",volumeName,pathString);
      }
      else
      {
         printf("FEMapVolumeAndDirectoryToPath failed: %x\r\n",
                NetWareErrno);
      }
   }
}

/*
  This EVENT only supports a ReportProcedure.  The procedure gets
  called every time a trustee is added or removed to/from the file
  system.
*/

void ReportProcedure(LONG parameter)
{

   /*  Save the data in our local structure */

   etc = *(struct EventTrusteeChangeStruct *)parameter;

   /*  Wake up the main thread */

   SignalLocalSemaphore(semaphoreHandle);
}


/*  This procedure is called when the NLM is unloaded  */

void UnloadProcedure(void)
{
   printf("Inside UnloadProcedure() Procedure\r\n");

   if(UnregisterForEvent(eventHandle) != NULL)
      printf("Error during Unregister of event TRUSTEE_CHANGE\r\n");
   CloseLocalSemaphore(semaphoreHandle);
}

/*  Convert a NetWare-Style path to a DOS-Style path, in place */

void NetWarePathToDOSPath(BYTE *psp, LONG pathCount)
{
   LONG    componentLen;

   componentLen = *psp;           /* remember the length     */
   *psp++ = ':';                  /* put volume separator    */
   psp += componentLen;           /* get to next component   */

   while( -pathCount )
   {
      componentLen = *psp;        /* remember length         */
      *psp = '/';                 /* put directory separator */
      psp += componentLen + 1;    /* to next component       */
   }
   *psp = NULL;                   /* put null terminator     */
}
```

1.1.7 NLM Hello World: Example

```
NOTE: taken from HELLO.C in the \EXAMPLES directory


/***********************************************************************
   HELLO.C
************************************************************************

   This module shows how to write a "Hello, world" NLM. To run this
   NLM, simply load HELLO.NLM on the server, and it will print the
   message "Hello, world" on the screen.

   HELLO.C uses symbols in CLIB.IMP, THREADS.IMP, and NLMLIB.IMP.

***********************************************************************/

#include <nlm/stdio.h>

void main(void)
{
   printf( "Hello, world\n" );
}
```

1.1.8 Using atexit() functions: Example

NOTE: taken from ATEXIT.C in the \EXAMPLES directory

```
/*********************************************************************
   ATEXIT.C
**********************************************************************

  This example NLM illustrates the use of atexit function(s).
  An atexit() function is one that is called by CLib AFTER the
  NLM's threads have been destroyed.  This is not a workable
  solution for NLMs that use stack-based resources which are
  hidden from the atexit() functions.  For those types of
  situations, you would want to use a signal handler, so you
  can "tell" the threads to shutdown themselves.  See the SIGTERM
  example for an illustration of signal handling.

  notes:  1.  The atexit() functions are always called when the NLM
              is exiting, unless the NLM is exiting via abort().
              Remember that the SIGABRT signal calls abort()!
         2.  All of your NLM's threads have been destroyed BEFORE any
              of these functions are called.
          3.  You can call CLib functions from your atexit functions.
              You must only free resources, however, don't try and
              allocate new resources!  i.e. don't try to open files,
              or service queue jobs, or start new threads, ...
          4.  Your screen is still intact, so you can use any of the
              standard library's IO functions here. i.e. printf,...

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/nwtypes.h>
#include <nlm/nwsemaph.h>
#include <nlm/nwerrno.h>

FILE    *myOpenFile;  /*  file to open       */
LONG    mySemaphore;  /*  semaphore to open  */
char    *myMemPtr;    /*  memory to allocate */

/*
  Following are the atexit functions to register.  They are called in
  a last-in first-out fashion, so if you will have dependancies
  keep that in mind.
*/

/*  This function closes my file, if it was opened. */

void CloseMyFile()
{
   if( myOpenFile != NULL )
   {
      fclose( myOpenFile );
      printf("myOpenFile was closed...\n");
   }
   else
   {
      printf("myOpenFile was not open...\n");
   }
}

/*  This function closes my semaphore, if it was opened. */

void CloseMySemaphore()
{
   if( mySemaphore != NULL )
   {
      CloseLocalSemaphore( mySemaphore );
      printf("mySemaphore was closed...\n");
   }
   else
   {
      printf("mySemaphore was not opened...\n");
   }
}

/*  This function frees my memory, if it was allocated. */

void FreeMyMemory()
{
   if( myMemPtr != NULL )
   {
      free( myMemPtr );
      printf("myMemPtr was freed...\n");
   }
   else
   {
      printf("myMemPtr was not allocated...\n");
   }
}

int main(void)
{
   if( (myMemPtr = malloc( 200 )) == NULL )
   {
      printf("\n  Could not allocate memory\n");
   }

   if( (mySemaphore = OpenLocalSemaphore( 1 )) == EFAILURE )
   {
      printf("\n  Could not open semaphore\n");
   }

   if((myOpenFile = fopen( "SYS:MYFILE", "r" )) == NULL )
   {
      printf("\n  Could not open my file\n");
   }

   if( atexit( CloseMyFile ) == EFAILURE )
   {
      printf("\n  Could not register CloseMyFile\n");
   }

   if( atexit( CloseMySemaphore ) == EFAILURE )
   {
      printf("\n  Could not register CloseMySemaphore\n");
   }

   if( atexit( FreeMyMemory ) == EFAILURE )
   {
      printf("\n  Could not register FreeMyMemory\n");
   }

   printf("  Okay, I’m ready.  Unload me or Press ESC\n");
   while(1)
   {
      if( getch() == 0x1b )
         break;    /* break on ESCAPE key */

      printf("\n\n  Press ESC to have the NLM unload itself...\n");
   }

   /*  When the NLM exits on its own, the registered atexit functions
       are called by CLib...  */

   return(0);
}
```

1.1.9 Using AtUnload() functions: Example

```
NOTE: taken from ATUNLOAD.C in the \EXAMPLES directory
```

```
/*********************************************************************
   ATUNLOAD.C
**********************************************************************

  This example NLM illustrates the use of an AtUnload function.
  An AtUnload function is one that is called by CLib AFTER the
  NLM’s threads have been destroyed.  This is not a workable
  solution for NLMs that use stack-based resources which are
  hidden from the AtUnload function.  For those types of
  situations, you would want to use a signal handler, so you
  can "tell" the threads to shutdown themselves.  See SIGTERM
  example for an illustration of signal handling.
  notes:  1.  The AtUnload function is called ONLY if the NLM is being
               unloaded from the command line.  Thus, you would most
              likely use this when your NLM will never exit on its own.
         2.  All of your NLM's threads have been destroyed BEFORE this
               function is called.
          3.  You can call CLib functions from your AtUnload function.
               You must only free resources, however, don't try and
               allocate new resources!  i.e. don't try to open files,
               or service queue jobs, or start new threads, ...
          4.  Your screen is still intact, so you can use any of the
               standard library's IO functions here. i.e. printf,...

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/malloc.h>
#include <nlm/nwtypes.h>
#include <nlm/nwthread.h>

char   *myMemPtr;     /* pointer to my memory */

/*
  This is the AtUnload function called by CLib after the NLM's
  threads are destroyed.  It should do all cleanup before returning,
  unless you are also using atexit functions.

  The function is registered via the CLib function AtUnload().  See the
  code in main().
*/

void NLMsAtUnloadFunction()
{
   if( myMemPtr != NULL )
      free( myMemPtr );
   printf("Example#2 NLM has cleaned up its resources...\n");
}

int main(void)
{
   if( (myMemPtr = malloc( 200 )) == NULL )
   {
      printf("\n\n\n  Could not allocate memory!\n\n");
      printf(" Go ahead an unload me from the command line
                anyway...\n\n");
   }

   AtUnload( NLMsAtUnloadFunction );     /* register unload function */
   printf("  Okay, I'm ready.  Unload me!\n");

   while(1)
   {
      if( getch() == 0x1b )
         break;    /* break on ESCAPE key */
      printf("\n\n  Press ESC to have the NLM unload itself...\n");
   }

   /*  If the NLM unloads itself, the AtUnload function is NOT called!
       So, I'll call it myself to free my resources... */

   NLMsAtUnloadFunction();

   return(0);
}
```

1.1.10 Using check functions: Example

NOTE: taken from CHECK.C in the \EXAMPLES directory

```
/*********************************************************************
   CHECK.C
**********************************************************************

  This example NLM illustrates the use of a CHECK function.
  A CHECK function is one that is called by NetWare BEFORE
  an NLM is unloaded.  It allows the function to determine
  if it is "safe" to unload the NLM, and then convey that
  information to the console operator via the NetWare
  System Console screen.

  To generate the warning, enter a 1 (non-zero value) at the
  command line:

     LOAD CHECK 1

  To skip the warning, enter a 0 at the command line:

     LOAD CHECK 0

  notes:  1.  The function returns non-zero to tell NetWare to
              issue a warning before the NLM is unloaded.
          2.  The function returns zero to tell NetWare that it
              is okay for the NLM to be unloaded.
          3.  The function should execute quickly, and should
              NOT relinquish control.
          4.  The function should NOT call any CLib functions.
          5.  Messages can be printed using ConsolePrintf
          6.  Check functions will not work properly if they have a
              program offset of zero.  To avoid this problem, don't
              list the check function first in your program (for
              instance, this example declares main() before
              CheckFunction() ).
          7.  To build a makefile for this example using QMK386,
              use the following syntax to register the check
```

```
                 function through the linker:

                    QMK386 CHECK /ocCheckFunction

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <ntypes.h>
#include <nlm/nwtypes.h>
#include <nlm/nwconio.h>

int  NLMisBusyRightNow = 0;

/*
  This is the CHECK function.  It is called by NetWare before the NLM
  is unloaded.  It returns non-zero to signal the OS to prompt the
  operator before unloading.  It returns zero to tell the OS it is okay
  to unload the NLM.
*/

int main(int argc, char *argv[])
{
   if( argc > 1 )
      NLMisBusyRightNow = atoi(argv[1]);   /* pick up flag */

   printf("\r\n\n\n  The value of NLMisBusyRightNow is %d\n",
NLMisBusyRightNow);
   printf("\n\n  Go ahead and unload it from the command line...\n");

   while(1)
   {
      if( getch() == 0x1b )
         break;    /* break on ESCAPE key */

      printf("\n\n  Press ESC to have the NLM unload itself...\n");
   }
   /* Returning from the last thread of execution will unload
      the NLM */

   return(0);
}

int CheckFunction()
{
   ConsolePrintf("\n CheckFunction called.\n");

   if( NLMisBusyRightNow != 0 )
   {
      ConsolePrintf("NLM currently busy!  Do not unload it
                     right now!\r\n");
      return(1);
   }
   return(0);
}
```

1.1.11 Using check functions: Example 2

NOTE: taken from UNLOADNO.C in the \EXAMPLES directory

```
/*********************************************************************
   UNLOADNO.C
**********************************************************************

   This module demonstrates how to prevent your NLM from being
   unloaded. It uses a check function to accomplish this task.

   Check functions such as NoUnload are registered through the linker.
   Use the following syntax with QMK386 to generate a makefile that
   registers this check function:

      QMK386 UNLOADNO /ocNoUnload

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/nwtypes.h>
#include <nlm/nwconio.h>
#include <nlm/nwthread.h>

/*  The following check function is called by NetWare when
    someone attempts to unload my NLM.  It will ungetch() an 'n' onto
    the system console, and return non-zero to request that the OS
    prompt the operator before unloading.  Since the 'n' has been
    pushed onto the console, it will automatically return to the prompt
    without allowing the operator to unload the NLM.  Make sure and
    allow for some other mechanism for unloading, or your NLM will
    remain loaded until the server is downed.
*/

static LONG   MyTGID;

int main(void)
{
   MyTGID = GetThreadGroupID();

   printf("Go ahead and try to unload this NLM!\n");
   printf("<Press any key to exit from here>\n");
   getch();

   return(0);
}

int NoUnload(void)
```

```
{
   LONG   OldTGID;
   LONG   OldScrID;
   LONG   NewScrID;

   /*  Establish context for this thread. */

   OldTGID  = SetThreadGroupID(MyTGID);
   OldScrID = GetCurrentScreen();
   NewScrID = CreateScreen("System Console",0);

   /*  Make sure the current screen is the system console */

   if( OldScrID != NewScrID)
      SetCurrentScreen(NewScrID);

   /*  push an ’n’ onto the console */

   ungetch(’n’);

   /*  Reset the screen ID if necessary  */

   if( OldScrID != NewScrID)
      SetCurrentScreen(OldScrID);

   DestroyScreen(NewScrID);
   SetThreadGroupID(OldTGID);

   return(1);  /*  ask NetWare to supply a warning message */
}
```

1.1.12 Using signal handlers: Example

NOTE: taken from SIGTERM.C in the \EXAMPLES directory

```
/**********************************************************************
   SIGTERM.C
**********************************************************************

   This example NLM illustrates the use of signal handlers.  Signal
   handlers provide a means of performing a graceful shutdown in the
   NLM environment.  The big difference between signal handlers and
   AtUnload or atexit functions is that signal handlers are called
   BEFORE the NLM’s threads are destroyed.  This way, you can have
   the threads shut down themselves.

  notes:  1.  This NLM will illustrate how to handle the SIGTERM
              signal.
              This signal is raised by CLib if the NLM is unloaded
              from the command line.
          2.  If your NLM exits on its own, YOU must raise SIGTERM!
          3.  You can call CLib functions from your signal handler.
          4.  Your screen is still intact, so you can use any of the
              standard library's IO functions here. i.e. printf,...
          5.  You can relinquish control.

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <ntypes.h>
#include <nlm/signal.h>
#include <nlm/nwtypes.h>
#include <nlm/nwthread.h>
#include <nlm/nwerrno.h>

int     ThreadCounter;   /* the number of threads that are running
*/
int     ShutDownFlag;    /* set to TRUE by the SIGTERM signal handler
*/

/*
  Following is the function that I will register for handling the
  SIGTERM signal.  It will be called by CLib if the NLM is unloaded,
  or by myself if you hit the escape key...
*/

#pragma off(unreferenced);
void MySignalHandler(int sigtype) /* sigtype is SIGTERM, SIGABRT, ...
*/
#pragma on(unreferenced);
{
   ShutDownFlag = TRUE;    /* tell the threads to shutdown */

   printf("Inside signal handler, waiting for threads to stop...\n");

   while( ThreadCounter > 0 )
   {
      delay( 500 );        /* wait half a second...        */
   }

   printf("Inside signal handler, threads have stopped...\n");
}

/*
  This is the thread running. More than one instance may be running.
*/

void MyThread( LONG *AmtOfMem )
{
   char    *myMemPtr;
   int     ThreadID = GetThreadID();
   ++ThreadCounter;
   if( (myMemPtr = malloc( *AmtOfMem )) == NULL )
   {
      printf("\n  Thread %08x: Could not allocate memory!\n",
              ThreadID);
   }

   printf("  Thread %08x:  Waiting...\n", ThreadID);
   while( ShutDownFlag != TRUE )
   {
      ThreadSwitch();   /* just spin here... */
   }

   if( myMemPtr != NULL ) free( myMemPtr );
   printf("  Thread %08x:  Exiting...\n", ThreadID);
   -ThreadCounter;
}

main(int argc, char *argv[])
{
   LONG   amountOfMemory = 200;
   int    numThreads = 3,i;

   if( argc > 1 )
      amountOfMemory = atoi( argv[1] );

   printf("memory each thread will allocate = %d\n",amountOfMemory);

   for( i = 0; i < numThreads; ++i )
   {
     if( BeginThread(MyThread,NULL,NULL,&amountOfMemory) == EFAILURE )
      {
         printf("Could not start thread number %d\n",i);
      }
   }

   signal(SIGTERM, MySignalHandler);   /* register signal handler */

   printf(" Okay, the threads have started.  Unload me or Press
            ESC!\n");
   while(1)
   {
      if( getch() == 0x1b )
         break;    /* break on ESCAPE key */

      printf("\n\n  Press ESC to have the NLM unload itself...\n");
   }

   /*  When the NLM exits on its own, YOU must raise SIGTERM! */
   raise(SIGTERM);
}
```

1.1.13 Using access(): Example

NOTE: taken from ACCESS.C in the \EXAMPLES directory

```
/*********************************************************************
  ACCESS.C
*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/string.h>
#include <nlm/unistd.h>
#include <nlm/nwerrno.h>
#include <nlm/errno.h>
#include <nlm/sys/stat.h>

int main(void)
{
   int    rc;
   char   filename[200];
   struct stat statblk;

   errno = 0;

   printf("enter filename: ");
   gets(filename);

   rc = access(filename, W_OK | R_OK);
   if(rc)
   {
      printf("access returned %d\n", rc);
      printf("%s\nNetWareErrno=%d\n\n", strerror(errno),
             NetWareErrno);
     exit(1);
   }

   rc = stat(filename, &statblk);
   if(rc)
   {
      printf("stat returned %d\n", rc);
      printf("%s\nNetWareErrno=%d\n\n", strerror(errno),
             NetWareErrno);
     exit(1);
   }

   printf("st_dev = %d\n",statblk.st_dev);
   printf("st_ino = %d\n",statblk.st_ino);
   printf("st_mode = %hu\n",statblk.st_mode);
   printf("st_nlink = %hd\n",statblk.st_nlink);
   printf("st_uid = %d\n",statblk.st_uid);
   printf("st_gid = %hd\n",statblk.st_gid);
   printf("st_rdev = %d\n",statblk.st_rdev);
   printf("st_size = %d\n",statblk.st_size);
   printf("st_atime = %d\n",statblk.st_atime);
   printf("st_mtime = %d\n",statblk.st_mtime);
   printf("st_ctime = %d\n",statblk.st_ctime);
   printf("st_btime = %d\n",statblk.st_btime);
   printf("st_attr = %d\n",statblk.st_attr);
   printf("st_archivedID = %d\n",statblk.st_archivedID);
   printf("st_updatedID = %d\n",statblk.st_updatedID);

   return(0);
}
```

1.1.14 Using tmpnam: Example

```
NOTE: taken from TEMPNAME.C in the \EXAMPLES directory
```

```
/*********************************************************************
  TEMPNAME.C
**********************************************************************

  This example demonstrates the tmpnam() function.

  OUTPUT:

  The new name in 'myBuffer' is: _T-00001.TMP
  The new name in the 'static buffer' is: _T-00002.TMP
  The name that 'ptr' points to is: _T-00003.TMP

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>

int main(void)
{
   char myBuffer[L_tmpnam];
   char *ptr;

   tmpnam(myBuffer);
   printf("The new name in 'myBuffer' is: %s\n",myBuffer);

   ptr = tmpnam(NULL);
   printf("The new name in the 'static buffer' is: %s\n", ptr);

   ptr = (char *) malloc(L_tmpnam);
   tmpnam(ptr);
   printf("The name that 'ptr' points to is: %s\n", ptr);
   free(ptr);

   return(0);
}
```

1.1.15 Using _makepath and _splitpath: Example

```
/*********************************************************************

    This example demonstrates the functions _makepath and _splitpath.

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/nwfileio.h>

int main(void)
{
char   full_path[_MAX_PATH];
char   vol[_MAX_VOLUME];
char   dir[_MAX_DIR];
char   fname[_MAX_FNAME];
char   ext[_MAX_EXT];
_makepath(full_path,"SYS","acct\\dueacct\\","test","c");
printf("\nFULL PATH: \n  %s\r\n",full_path);
 _splitpath(full_path,vol,dir,fname,ext);
 printf("\nCOMPONENTS AFTER _splitpath\r\n");
 printf("  vol : %s\r\n",vol);
 printf("  dir   : %s\r\n",dir);
 printf("  fname : %s\r\n",fname);
 printf("  ext   : %s\r\n",ext);
 return(0);
 }
```

1.1.16 Using unlink(): Example

```
NOTE: taken from UNLINK.C in the \EXAMPLES directory


/**********************************************************************
  UNLINK.C
**********************************************************************

  This example demonstrates how to delete a file using unlink().

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/unistd.h>
```

```
int main(int argc, char *argv[])
{
   if(argc != 2)
   {
      printf("\nusage: UNLINK <filename>");
     exit(1);
   }

   if(unlink(argv[1]))
      printf("\nCOULD NOT DELETE: %s", argv[1]);
   else
      printf("\nDELETED FILE:  %s", argv[1]);

   return(0);
}
```

1.1.17 Using readdir(): Example

```
NOTE: taken from DIRECTRY.C in the \EXAMPLES directory
```

```
/**********************************************************************
  DIRECTRY.C
**********************************************************************

  USAGE:   DIRECTRY <PATH>

  EXAMPLE: DIRECTRY SYS:\EXAMPLES\*.*

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/errno.h>
#include <nlm/dirent.h>

int main(void)
{
   char  path[255];
   int   cnt = 0;
   DIR   *dirStructP;
   DIR   *direntp;

   printf("Enter the path to do readdir() on: ");
   scanf("%s", path);

   if((dirStructP = opendir(path)) != NULL)
   {
      while((direntp = readdir(dirStructP)) != NULL)
      {
         cnt++;
         /* possibly dump out file info here...*/
      }
      printf("%d entries scanned\n", cnt);
      closedir(dirStructP);
   }
   else
   {
      printf("opendir returned errno: %#x\n", errno);
     exit(1);
   }

   return(0);
}
```

1.1.18 NCP Extension Server: Example

```
NOTE: taken from ECHOSERV.C in the \EXAMPLES directory


/
**********************************************************************
   ECHOSERV.C
**********************************************************************

  This module illustrates how to create an NLM Server that uses
  the NCP Extension APIs in CLib.  ECHOSERV.C is NLM-specific but its
  counterpart ECHOCLNT.C can be run from a server or a workstation.

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/nwncpx.h>
#include <nlm/nwadv.h>
#include <nlm/nwthread.h>
#include <nlm/nwconio.h>
#include <nlm/nwerrno.h>

/* Prototypes */

BYTE EchoServer(NCPExtensionClient *client, BYTE *requestData,
                LONG requestDataLen, BYTE *replyData,
                LONG *replyDataLen);

void EchoServerConnDownHandler(LONG connection, LONG eventType);

/* Globals */

int  myThreadGroupID;
char NCPExtName[] = "ECHO SERVER";
```

```
struct queryDataStruct
{
   LONG CharsEchoed;
   LONG ErrorsServicingRequests;
   LONG unused[6];
} *queryData;

int main(void)
{
   int rc;

   myThreadGroupID = GetThreadGroupID();

   ConsolePrintf("Registering NCP Extension: %s\n", NCPExtName);

   if(SetThreadContextSpecifier(GetThreadID(), NO_CONTEXT)
     != ESUCCESS)
   {
      ConsolePrintf("Error doing pre-registration processing, %s not
             loaded\n", NCPExtName);
      return 1;
   }

   rc = NWRegisterNCPExtension(NCPExtName, EchoServer,
                         EchoServerConnDownHandler, NULL, 1, 0, 0,
                        &queryData);
   if (rc)
   {
      ConsolePrintf("Error %d registering NCP Extension: %s\n", rc,
                    NCPExtName);
      return 2;
   }

   queryData->CharsEchoed             = 0;
   queryData->ErrorsServicingRequests = 0;

   printf("Press any key to unload echo server.\n");
   getch();

   rc = NWDeRegisterNCPExtension(queryData);
   if(rc)
   {
      ConsolePrintf("Error %d Deregistering NCP Extension: %s\n",
                     NCPExtName);
      return 3;
   }

   ConsolePrintf("NCP Extension %s Deregistered\n", NCPExtName);
   return 0;
}


/*  A note about checking parameters:  These requests are coming
```

```
    over the wire and little is known about the entity formulating
    them.  It is up to the developer to decide how much to trust
    the incoming requests, this example checks the length of the
    parameters requestDataLen and replyDatalen.  */

BYTE EchoServer(NCPExtensionClient *client, BYTE *requestData,
                LONG requestDataLen, BYTE *replyData, LONG
*replyDataLen)
{
   int savedThreadGroupID;
   int rc;

   if(requestDataLen < 1)
   {
      /* Expect exactly 1, there's a problem if it's 0 */

     ConsolePrintf("%s reports bad parameters on call from client at "
                    "connection: %ld, task: %ld\n",
                    client->connection, client->task);

      queryData->ErrorsServicingRequests++;
      return 0x5F;
   }

   savedThreadGroupID = GetThreadGroupID();
   if(SetThreadGroupID(myThreadGroupID) == EFAILURE)
   {
      queryData->ErrorsServicingRequests++;
      return 0x5E;
   }

   if (putchar(*(char *)requestData) == EOF)
   {
     ConsolePrintf("%s couldn't echo char to screen!\n", NCPExtName);

      queryData->ErrorsServicingRequests++;
      rc = 0x5D;
      goto exit0;
   }

   /* Check the length of the reply data buffer since we don't have
      control over the caller. (Shouldn't write to memory when unsure
      of the size.)   */

   if (*replyDataLen < 1)
   {
      /*  Expect exactly 1, there's a problem if it's 0 */

     ConsolePrintf("%s reports bad parameters on call from client at "
                    "connection: %ld, task: %ld\n",
                    client->connection, client->task);

      queryData->ErrorsServicingRequests++;
      rc = 0x5C;
      goto exit0;
   }

   *replyDataLen = 1;
   *replyData = *requestData;
   queryData->CharsEchoed++;
   rc = 0;

exit0:

   if(SetThreadGroupID(savedThreadGroupID) == EFAILURE)
      ConsolePrintf("%s reports error restoring callers thread group
                     context\n", NCPExtName);
   return rc;
}

void EchoServerConnDownHandler(LONG connection, LONG eventType)
{
   char *eventString;

   /*  NOTE:  No context is needed in this callback, so we don't set
       the thread group ID.     */

   switch(eventType)
   {
      case CONNECTION_BEING_RESTARTED:
     {
         eventString = "connection being restarted";
         break;
     }
      case CONNECTION_BEING_KILLED:
     {
         eventString = "connection being killed";
         break;
     }
      case CONNECTION_BEING_LOGGED_OUT:
     {
         eventString = "connection being logged out";
         break;
     }
      case CONNECTION_BEING_FREED:
     {
         eventString = "connection being freed";
         break;
     }
      default:
     {
         eventString = "unknown event type";
     }
   }
   ConsolePrintf("\n%s got notification of connection event on
          conn #%d\n" "event = %s\n", NCPExtName, connection,
          eventString);
```

```
}
```

1.1.19 Creating a multiple-loadable NLM: Example

NOTE: taken from RENTRANT.C in the \EXAMPLES directory

```
/**********************************************************************
   RENTRANT.C
**********************************************************************

  RENTRANT.C demonstrates the use of an NLM as a multiple-loadable
  daemon which can perform a task as scheduled with different copies
  of the NLM running continually in the background.

  This example NLM illustrates the use of a START function.

  The following command uses QMK386 to generate a Watcom makefile for
  RENTRANT.C:

     QMK386 RENTRANT /nr /x /osMultipleLoadFilter

     /nr                   = ReEntrant
     /x                    = no default screen for NLM
     /osMultipleLoadFilter = register start function

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/string.h>
#include <ntypes.h>
#include <nlm/nwthread.h>
#include <nlm/nwconio.h>
#include <nlm/time.h>

#define kMaxArguments         12
#define SECONDS_SINCE_START   clock() / 100

#define kUsage  "\nUsage:\n"\
                "Rentrant <perform-interval>\n"\
                "perform-interval (every) [[days:]hours:]minutes\n\n"

/* Globals */

typedef struct
{
   int      Days;
   int      Hours;
   int      Minutes;
   int      Executions;
   clock_t  TotalSeconds;
} Globals;

typedef struct resource_list
{
   struct resource_list *next;
   int    screenHandle;
} ResourceList;

typedef void   (*PVF) ( void *);
int            gAlreadyLoaded = 0;
int            gMainThreadGroupID;
char           *gProgramName = "Rentrant";
Globals        *gG;
ResourceList   *gResList = (ResourceList *) NULL;
typedef struct LoadDefinitionStructure *LoadDefStructPtr;
typedef struct ScreenStruct *ScreenStructPtr;

/* Prototypes */

void   StrCpy(register char *t, register char *s);
int    ParseCommandLine(char *commandLine, int *argc, char *argv[]);
void   DisplayPerformances(Globals *g);
void   DisplayTimeLeft(clock_t timeLeft, int doZero);
int    LogScreenHandle(int scrH);
void   CleanupResource(int which);
void   Cleanup(void);
extern LONG   _Prelude( );
LONG   MultipleLoadFilter(LoadDefStructPtr NLMHandle,
                          ScreenStructPtr    initErrorScreenID,
                          BYTE               *cmdLineP,
                          BYTE               *loadDirPath,
                          LONG               uninitDataLen,
                          LONG               NLMFileHandle,
                          LONG               cdecl (*readFunc)());

/*********************************************************************
 **  Main Routine
 */

void main(int argc, char *argv[])
{
   int      firstTime, first, second, third;
   int      scrH, myThreadGroupID;
   char     **argV;
   char     *args[kMaxArguments];
   char     commandLine[200+1];
   char     threadName[17+1+13];
   char     ch;
   Globals  g;
   register clock_t   whenToExecuteNext;
   register int i;

   memset(&g, 0, sizeof(Globals));
```

```
gG = &g;    /*  for debugging...  */

if(!gAlreadyLoaded)
{
   gMainThreadGroupID = GetThreadGroupID();
   RenameThread(gMainThreadGroupID, "Rentrant-main");
   gAlreadyLoaded = 1;
   firstTime      = TRUE;
   argV           = argv;
   AtUnload(Cleanup);
}
else
{
   sprintf(threadName, "Rentrant-#%d", gAlreadyLoaded);
   myThreadGroupID = GetThreadGroupID();
   RenameThread(myThreadGroupID, threadName);
   gAlreadyLoaded++;
   firstTime = FALSE;
   argV      = args;
}

scrH = CreateScreen("Sample Reentrant NLM", 0);
if(!scrH)
{
   ConsolePrintf("\nUnable to create screen...");
   goto NoScreenExit;
}

LogScreenHandle(scrH);
SetCurrentScreen(scrH);
printf("\nSample Reentrant NLM: %d\n", gAlreadyLoaded);

if(!firstTime)  /*  command line must be separately parsed...  */
{
   argV[0] = gProgramName;

  /*  as passed to BeginThreadGroup() */
   StrCpy(commandLine, (char *) argc);

   if(ParseCommandLine(commandLine, &argc, argV+1))
   {
      ConsolePrintf("%c", 0x07);
      goto UsageError;
   }

   argc++;  /*  to compensate for missing program name...  */

   ConsolePrintf("\nCommand:\n");

   /*  debugging only, can remove...  */
   for (i = 0; i < argc; i++)
      ConsolePrintf("%s ", argV[i]);
}
```

```
   if(argc < 2)
   {
UsageError :
      ConsolePrintf(kUsage, argV[0]);
      goto Exit;
   }

   first = second =   third = -1;

   sscanf(argV[1], "%d:%d:%d", &first, &second, &third);

   if(first != -1 && second != -1 && third != -1)
   {
      g.Days    = first;
      g.Hours   = second;
      g.Minutes = third;
   }
   else if(first != -1 && second != -1)
   {
      g.Hours   = first;
      g.Minutes = second;
   }
   else      /*  'first' must be nonzero  */
   {
      g.Minutes = first;
   }

/*
** Loop awaiting keyboard interrupt to halt this daemon. Yield to
** server very frequently. Wake up to perform work periodically...
*/
   gotoxy(0, 23);
   printf("Press 'q' at any time to halt this daemon...");

   g.TotalSeconds = g.Days    * 24 * 60 * 60
                  + g.Hours   * 60 * 60
                  + g.Minutes * 60;

   whenToExecuteNext = g.TotalSeconds;

   while(TRUE)
   {
      if(whenToExecuteNext < SECONDS_SINCE_START)
      {
         whenToExecuteNext += g.TotalSeconds;
         DisplayTimeLeft(0L, TRUE);

         /*  perform work stuff here...  */
         {
            gotoxy(0, 5);
            printf("Performing...");
            for (i = 0; i < 100000; i++);
            gotoxy(0, 5);
            printf("             ");
         }

         g.Executions++;
         DisplayPerformances(&g);
      }

      if(kbhit())
      {
         ch = getch();
         if (ch == ’q’ || ch == ’Q’)
            break;
      }

      DisplayTimeLeft(whenToExecuteNext - SECONDS_SINCE_START, FALSE);
     ThreadSwitchLowPriority();
   }

Exit :
   CleanupResource(scrH);

NoScreenExit :
   ExitThread(EXIT_THREAD, 0);
}

void StrCpy(register char *t, register char *s)
{
   while (*t++ = *s++);
}

LONG MultipleLoadFilter(LoadDefStructPtr   NLMHandle,
                        ScreenStructPtr    initErrorScreenID,
                        BYTE               *cmdLineP,
                        BYTE               *loadDirPath,
                        LONG               uninitDataLen,
                        LONG               NLMFileHandle,
                        LONG               cdecl (*readFunc)())
{
   int   myThreadGroupID;

   if(!gAlreadyLoaded)            /*  first time through  */
      return _Prelude(NLMHandle, initErrorScreenID, cmdLineP,
                      loadDirPath, uninitDataLen, NLMFileHandle,
                      readFunc);

   /*  subsequent times through...  */
   myThreadGroupID = SetThreadGroupID(gMainThreadGroupID);
   BeginThreadGroup((PVF) main, NULL, NULL, cmdLineP);
   SetThreadGroupID(myThreadGroupID);

   return 0L;
}

/*  into argc/argv format...  */
```

```
int ParseCommandLine(char *commandLine, int *argc, char *argv[])
{
   register char   *p = commandLine;

   if (!p)
      return -1;

   *argc = 0;

   while(TRUE)
   {
     argv[(*argc)++] = p;      /*  register last argument found... */

      while (*p && *p != ’ ’)
         p++;

      if (!*p)
         break;

      *p++ = ’\0’;
   }
   return 0;
}

void DisplayPerformances(Globals   *g)
{
   gotoxy(0, 3);
   printf("%d        ", g->Executions);
}

/* in seconds */
void DisplayTimeLeft(clock_t timeLeft, int doZero)
{
   clock_t   days, hours, minutes;

   if (doZero)
   {
      days = hours = minutes = 0;
      goto Display;
   }

   /*
   ** Remove days, hours and finally minutes from 100ths of seconds
   ** left
   */
   days      = timeLeft   / 86400;  /*  seconds per day     */
   timeLeft -= days       * 86400;
   hours     = timeLeft   / 3600;   /*  seconds per hour    */
   timeLeft -= hours      * 3600;
   minutes   = timeLeft   / 60;     /*  seconds per minute  */

Display :
   gotoxy(0, 2);
   printf("Next performance: %02d day", days);
   if (days != 1)
      putchar('s');

   putchar(' ');

   printf("%02d hour", hours);

   if (hours != 1)
      putchar('s');

   putchar(' ');

   printf("%02d minute", minutes);

   if (minutes != 1)
      putchar('s');
   else
      putchar(' ');
}

int LogScreenHandle(int scrH)
{
   register ResourceList   *r = gResList;

   if(!r)
   {
      r = gResList = (ResourceList *) malloc(sizeof(ResourceList));
      if (!r)
         return -1;
   }
   else
   {
      while (r->next)      /*  (find end of list...)  */
         r = r->next;

      r->next = (ResourceList *) malloc(sizeof(ResourceList));

      r = r->next;
      if (!r)
         return -1;
   }

   r->next         = (ResourceList *) NULL;
   r->screenHandle = scrH;

   return 0;
}

void CleanupResource(int which)
{
   register ResourceList   *next,
                           *prev = (ResourceList *) NULL,
                           *r = gResList;
   while(r)
   {
      next = r->next;

      if (r->screenHandle == which)
      {
         DestroyScreen(r->screenHandle);

         if (prev)         /*  patch holes left...  */
            prev->next = next;
         else
            gResList = next;

         free(r);
         break;
      }

      prev = r;
      r    = next;
   }
}

void Cleanup( void )
{
   register ResourceList   *next,
                           *r = gResList;

   while(r)
   {
      next = r->next;
      DestroyScreen(r->screenHandle);
      free(r);
      r = next;
   }
}
```

1.1.20 Salvaging Files: Example

```
NOTE: taken from SALVAGE.C in the \EXAMPLES directory


/**********************************************************************
   SALVAGE.C
**********************************************************************

  This NLM salvages files from a NetWare volume.  It also
  supports purging files.

**********************************************************************/

#include <nlm/stdio.h>
```

```
#include <nlm/nwtypes.h>
#include <nlm/nwtime.h>
#include <nlm/nwconio.h>
#include <nlm/nwfile.h>
#include <nlm/string.h>

int main(void)
{
   int    rc;
   int    commandChar,purgeAllFlag = 0;
   char   newFileName[100];
   char   fullPathName[200];
   struct _DOSTime filTim;
   struct _DOSDate filDat;
   char   scanDirectory[100], *charP;
   long   nen = - 1;
   DIR    dirP;

   printf("Directory to scan: ");
   gets(scanDirectory);

   if(!scanDirectory[0])
   {
      scanDirectory[0] = '\\';
      scanDirectory[1] = 0;
   }

   /* print screen header */
   clrscr();
   printf("  File Name      Size      Attr      Date  ");
   printf("     Time     Ser#   Vol#   Seq#\r\n");
   printf("————   ——-   ——   ——-");
   printf("   ——-   —-   —-   —-\r\n");
   gotoxy( 0, 24);
   printf("A - purge all;  P - purge; S - salvage;");
   printf(" <enter> - next file; X - exit" );
   SetScreenRegionAttribute( 24, 1, 0x70 );
   gotoxy( 0, 2);

   while (!ScanErasedFiles(scanDirectory, &nen, &dirP))
   {
      filTim = *(struct _DOSTime *) &dirP.d_time;
      filDat = *(struct _DOSDate *) &dirP.d_date;
      strcpy(newFileName,dirP.d_name);

      if ((charP = strchr( newFileName, '.' ) ) == NULL)
         charP = " ";
      else
         *charP++ = 0;  /* overwrite period with a zero */

      if (wherey() == 24)
      {
         ScrollScreenRegionUp( 2, 22 );
         gotoxy( 0, 23 );
```

```
 }

 printf("%-8.8s %-3.3s %8d   0x%04x   %02u/%02u/%02u",
    newFileName, charP, dirP.d_size, dirP.d_attr,
    filDat.month, filDat.day, filDat.yearsSince80+80);

 printf("   %02u:%02u:%02u   %4d   %4d   %4d\r\n",
    filTim.hour, filTim.minute, filTim.bisecond,
    dirP.d_ino, dirP.d_dev, nen & 0xFFFFFF);

/* underline current file, un-underline previous file */
 SetScreenRegionAttribute(wherey()-1, 1, 10 );
 SetScreenRegionAttribute(wherey()-2, 1, 7 );

 if(!purgeAllFlag)
 {
    commandChar = getch();
    if(commandChar == 'A')
       purgeAllFlag = 1;
 }

 if((commandChar == 'p') || (commandChar == 'P') || purgeAllFlag)
 {
    strcpy(fullPathName,scanDirectory);

    if(scanDirectory[1])
       strcat(fullPathName,"\\");

    strcat(fullPathName,dirP.d_name);

    if (rc = PurgeErasedFile( fullPathName, nen ))
       printf("Could Not Purge File %s;  error = %d\r\n",
              fullPathName,rc);
 }
 else if((commandChar == 's') || (commandChar == 'S'))
 {
    if (wherey() == 24)
    {
       ScrollScreenRegionUp( 2, 22 );
       gotoxy( 0, 23 );
    }

    printf("new file name: ");
    gets(newFileName);
    strcpy(fullPathName,scanDirectory);

    if(scanDirectory[1])
       strcat(fullPathName,"\\");

    strcat(fullPathName,dirP.d_name);
    if (rc = SalvageErasedFile( fullPathName, nen, newFileName))
       printf("Could Not Salvage File %s;  error = %d\r\n",
              dirP.d_name,rc);
 }
```

```
      else if((commandChar == 'x') || (commandChar == 'X'))
      {
         break;
      }
   }
   return(0);
}
```

1.1.21 Searching for Files: Example

```
NOTE: taken from WHEREIS.C in the \EXAMPLES directory


/*********************************************************************
   WHEREIS.C
*********************************************************************

  This module finds all files matching the given file specification.
  It accepts NetWare-style paths.

*********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/string.h>
#include <ntypes.h>
#include <nlm/nwconio.h>
#include <nlm/ctype.h>
#include <nlm/unistd.h>
#include <nlm/nwdir.h>
#include <nlm/dirent.h>

#define   skipspace( x )  while( isspace( *x ) ) ++x
#define   nextspace( x )  while( *x && !isspace( *x ) ) ++x
#define   CWS     0
#define   CWV     1
#define   CWP     2
#define   ALL     99

/* Globals */
extern      char   *GetWorkArea(void);
extern      char   *next_arg(char *);
char        fid[100];
static      breakkey = FALSE;

int main(int argc, char *argv[])
{
   int hscrn;

   if (argc < 2)
   {
      ConsolePrintf("\nUsage: WHEREIS <filename>\n");
      exit(1);
   }

   hscrn = CreateScreen("WHEREIS v1.0", AUTO_DESTROY_SCREEN);
   SetCurrentScreen(hscrn);
   DisplayScreen(hscrn);

   strupr(strcpy(fid, argv[1]));
   printf("WHEREIS: %s...\n",fid);

   dowhereis(fid);

   PressAnyKeyToContinue();

   return(0);
}

void dowhereis(char *s)
{
   char    dir[_MAX_PATH];
   char    fsv[_MAX_SERVER+_MAX_VOLUME+1];
   char    fdir[_MAX_PATH];
   char    fname[_MAX_FNAME],fext[_MAX_EXT],
both[_MAX_FNAME+_MAX_EXT];
   char    *p = next_arg(s); /* point at argument */

   if(!*p)
   {
      printf("No filename specified!");
      return;
   }

   strcpy(dir,GetWorkArea());

   /* get the file name specification */

   _splitpath(p,fsv,fdir,fname,fext);
   sprintf(both,"%s%s",strupr(fname),strupr(fext));

   breakkey = FALSE;

   /* startup the recursive file find operation */

   chdir(fsv);
   findit(both);
}

char *GetWorkArea(void)
{
   static  char   cwd[_MAX_PATH];
   static  char   serverName[_MAX_SERVER];
   static  char   volumeName[_MAX_VOLUME + 1];
   static  char   dirName[_MAX_DIR];
   if(getcwd(cwd,_MAX_PATH) == NULL)
      return NULL;

   ParsePath(cwd,serverName,volumeName,dirName);   /* shouldn’t fail!
*/

   return cwd;
}

char *next_arg(char *s)
{
   char    *p;

   skipspace(s);     /* ignore white */
   p = s;
   nextspace(s);     /* find next blank */
   *s = NULL;
   return(p);
}

static void findit(char *what)
{
   char dir[_MAX_PATH];
   DIR *dirStructPtr;
   DIR *dirStructPtrSave;

   getcwd(dir,_MAX_PATH);
   dirStructPtrSave = dirStructPtr = opendir(what);

   while(dirStructPtr && !breakkey)
   {
      dirStructPtr = readdir(dirStructPtr);
      if((dirStructPtr == NULL) || (dirStructPtr == -1))
         break;

      printf("  %s/%s\n",dir,dirStructPtr->d_name);
      if(kbhit() && getch() == 3)
         printf("^C\n",breakkey = TRUE);
   }

   if(dirStructPtrSave)
      closedir(dirStructPtrSave);

   /*  Now traverse the directories in this path */

   dirStructPtrSave = dirStructPtr = opendir("*.*");
   if(dirStructPtr == NULL)
      return;

   while(!breakkey)
   {
      dirStructPtr = readdir(dirStructPtr);
      if((dirStructPtr == NULL) || (dirStructPtr == -1))
```

```
         break;

      if(dirStructPtr->d_attr & _A_SUBDIR)
      {
         chdir(dirStructPtr->d_name);
         findit(what);
         chdir("..");
      }

      if(kbhit() && getch() == 3)
         printf("^C\n",breakkey = TRUE);
   }

   if(dirStructPtrSave)
      closedir(dirStructPtrSave);
}
```

1.1.22 Listing Registered NCP Extensions: Example

```
NOTE: taken from NCPSCAN.C in the \EXAMPLES directory


/**********************************************************************
  NCPSCAN.C
**********************************************************************

  This example demonstrates how to list registered NCP extensions using
  NWScanNCPExtensions (NLM).

**********************************************************************/

#include <nlm/stdio.h>
#include <nlm/stdlib.h>
#include <nlm/nwncpx.h>

int main(void)
{
   BYTE majorVersion, minorVersion, revision;
   char NCPExtensionName[MAX_NCP_EXTENSION_NAME_BYTES];
   int  cCode;
   LONG NCPExtensionID = BEGIN_SCAN_NCP_EXTENSIONS; /* -1 */

   do
   {
      cCode = NWScanNCPExtensions(&NCPExtensionID, NCPExtensionName,
                                  &majorVersion, &minorVersion,
                                  &revision, NULL);
      if(cCode == 0)
      {
         printf("\nExtension Name = %s, Extension ID = %lX",
                NCPExtensionName, NCPExtensionID);
         printf("\nMajor Version = %d, Minor Version = %d,
                Revision = %d\n", majorVersion, minorVersion,
                revision);
      }
   }
   while(cCode != -1);

   return(0);
}
```

1.1.23 Manipulating Extended Attribute Byte: Example

```
NOTE: taken from EXTATTRS.C in the \EXAMPLES directory


/
**********************************************************************
  EXTATTRS.C
**********************************************************************

  This example demonstrates how to manipulate the first attribute byte
  of an extended attribute.  EXTATTRS.C uses the functions
  GetExtendedFileAttributes and SetExtendedFileAttributes.

*********************************************************************/

#include <nlm/stdlib.h>
#include <nlm/stdio.h>
#include <nlm/nwfinfo.h>
#include <nlm/nwfileio.h>

int main(void)
{
   char    pn[300];
   BYTE    efa;

   printf("name: ");
   gets(pn);
   printf("efa: ");
   scanf("%x", &efa);

   printf("set rc = %d\r\n", SetExtendedFileAttributes(pn, efa));
   printf("get rc = %d\r\n", GetExtendedFileAttributes(pn, &efa));
   printf("new efa = %x\r\n", efa);

   return(0);
}
```

1.2 XPlat Examples

The following example illustrate the XPlat libraries. For additional examples, see NLM and NetWare Libraries for C (http://developer.novell.com/ndk/clib_sample.htm).

1.2.1 Hello World: Example

```
NOTE: taken from NWHELLO.C in the \EXAMPLES directory


/*********************************************************************
   NWHELLO.C
*********************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <ntypes.h>
#include <nwcaldef.h>
#include <nwapidef.h>
#include <nwclxcon.h>  /* NWCCGetPrimConnRef */
#include <nwmisc.h>    /* NWCallsInit        */

#ifndef N_PLAT_NLM
   NWCCODE        ccode;
   nuint32        connRef;
   NWCCConnInfo   returnInfo;
#endif

int main(void)
{
   printf("\n Hello World!");

   /* If this example is running as a client application (and not an
      NLM), perform a simple network task to determine if the
      environment is set up correctly.  Print the server name
      corresponding to the workstation’s primary connection */

#ifndef N_PLAT_NLM

   /* Initialize libraries */
   ccode = NWCallsInit(NULL, NULL);
   if(ccode)
   {
      printf("\nNWCallsInit returned %04X", ccode);
      exit(1);
   }

   /*  Find the workstation’s primary connection reference.  */
   ccode = NWCCGetPrimConnRef(&connRef);
   if(ccode)
   {
      printf("\nNWCCGetPrimConnRef returned %04X",ccode);
      exit(1);
   }

   /*  Get information about the primary connection. */
   ccode = NWCCGetAllConnRefInfo(connRef, NWCC_INFO_VERSION,
           &returnInfo);
   if(ccode)
   {
      printf("\nNWCCGetAllConnRefInfo returned %04X",ccode);
      exit(1);
   }

   /*  print the primary connection server name.  */
   printf("\n\n You are attached to server %s\n\n",
          returnInfo.serverName);

#endif

   return(0);
}
```

Revision History

A

The following table outlines all the changes that have been made to the NLM and NetWare Sample Code (in reverse chronological order):

Release Date	Revision Description
March 1, 2006	Updated format.
October 5, 2005	Transitioned to revised Novell documentation standards.
March 2, 2005	Fixed legal information.
February 16, 2004	Added a preface and divided the samples into two groups: CLib and XPlat.
September 2002	Fixed a problem in the _makepath and _splitpath example.
May 2002	Updated title page.
June 2001	Made changes to improve document accessibility.
May 2000	Added this revision history