Loading Keymaps

by Martin Taillefer

The Amiga uses a keymap to translate a raw key code from the keyboard into an escape sequence, a string, or an ANSI character value. The keymap helps make it possible to easily configure a single Amiga for use in different locales that use different keyboard layouts.

The Amiga system generally has a single global keymap used by the system software and by all applications. It is sometimes desirable to use keymaps in application software which differ from the current system keymap. This can be done by building a keymap into the application's code, or by loading alternate keymaps from disk.

On page 813 of the *Amiga ROM Kernel Reference Manual: Libraries, 3rd Edition*, the ''Setting the Default Keymap'' section states:

> ''When making a keymap the system default, first check whether the keymap has been loaded previously by checking the keymap list of the keymap.resource. If it has not been loaded already, it can be loaded from *DEVS:Keymaps* and added to the keymap list of the keymap.resource.''

Unfortunately, the manual doesn't mention how to perform either of these tasks. If you look through the existing Amiga documentation, you will have a very hard time finding any information at all on the keymap.resource (even though it was introduced to the OS in Release 1.2). Since the keymap.resource doesn't do very much and since it was not intended to be accessible to application programmers, no significant documentation was ever written. The keymap.resource only serves as an anchor for the system's list of keymaps. The keymap.resource has no mechanisms to manipulate its list if keymaps, so the application must perform this task.

The source code included with this article provides an easy to use function that loads a keymap from disk into memory and returns a pointer to it. The function adds the loaded keymap to the system's keymap list.

Note that applications rarely need to load an entire keymap from disk. In many cases, an application is better off duplicating the system's default keymap and making some minor modifications to it. The application can use this keymap privately via the console device, so the application doesn't have to interfere with the user's global keymap settings. See the article, ‘‘Customizing the Keypad Keymap’’ in the May/June 1993 issue of Amiga Mail.

Prior to V38 (2.04 and earlier), keymaps were kept in the *DEVS:Keymaps* directory. Starting with V38 (2.1), keymaps are kept in the KEYMAPS: multi-assign directory. This newer approach allows keymap files to reside in multiple independent directories all referenced via the multi-assign. This is similar to the way LIBS: and DEVS: are set up.

Another change that occurred in V38 was that the Input preferences editor now lets the user pick the keymap to use for the system. This is a very simple and natural choice for the user. In prior releases, the user was expected to fiddle with the *SetMap* program, which was not very intuitive. Starting with V38, the *SetMap* program is no longer a part of the Amiga system software distribution.

The LoadKeyMap() function listed below takes a single parameter which is the name of the keymap to load. This should include the full path to the keymap you wish to load. The function then checks to see if the keymap is already on the keymap.resource's keymap list. If this is the case, the function returns a pointer to the keymap that is already loaded. If the keymap is not in memory, then the function attempts to load the keymap from disk and install it in the system. The function returns `NULL` in case of any failure.

Unlike most other Amiga system resources, keymaps cannot be removed from memory once loaded. There is no open count maintained for keymaps. Keymaps are so small however, such a limitation is not a problem. This is why the sample code has no need for an UnloadKeyMap() function.

LoadKeyMap.h

```
#ifndef LOADKEYMAP_H
#define LOADKEYMAP_H

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

#ifndef EXEC_TYPES_H
#include <exec/types.h>
#endif

#ifndef DEVICES_KEYMAP_H
#include <devices/keymap.h>
#endif

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

struct KeyMap *LoadKeyMap(STRPTR name);

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

#endif /* LOADKEYMAP_H */
```

LoadKeyMap.c

```
#include <exec/types.h>
#include <exec/memory.h>
#include <exec/libraries.h>
#include <devices/keymap.h>
#include <dos/dos.h>

#include <clib/exec_protos.h>
#include <clib/utility_protos.h>
#include <clib/dos_protos.h>

#include <pragmas/exec_pragmas.h>
#include <pragmas/utility_pragmas.h>
#include <pragmas/dos_pragmas.h>

#include "loadkeymap.h"


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


extern struct Library *SysBase;
extern struct Library *UtilityBase;
extern struct Library *DOSBase;


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


/* Case-insensitive version of FindName() */
static struct Node *FindNameNC(struct List *list, STRPTR name)
{
struct Node *node;
WORD         result;

    node = list->lh_Head;
    while (node->ln_Succ)
    {
        result = Stricmp(name,node->ln_Name);
        if (result == 0)
            return(node);

        node = node->ln_Succ;
    }

    return(NULL);
}


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


struct KeyMap *LoadKeyMap(STRPTR name)
{
BPTR                   segment;
struct KeyMapResource *kr;
struct KeyMapNode     *kn;
STRPTR                 base;

    kn      = NULL;
    segment = NULL;

    /* open the keymap resource, in order to gain access to the keymap list */
    if (kr = (struct KeyMapResource *)OpenResource("keymap.resource"))
    {
        segment = NULL;
        base = FilePart(name);

        /* must access the list under Forbid() */
        Forbid();
```

```
        /* is the keymap we want already on the keymap list? */
        if (!(kn = (struct KeyMapNode *)FindNameNC(&kr->kr_List,base)))
        {
            /* if not on the keymap list, try loading it */
            if (segment = LoadSeg(name))
            {
                /* see if someone added it to the keymap list while we were
                 * doing a LoadSeg() (which broke Forbid() )
                 */
                if (!(kn = (struct KeyMapNode *)FindNameNC(&kr->kr_List,base)))
                {
                    kn = (struct KeyMapNode *)((segment << 2) + sizeof(BPTR));

                    /* we've loaded a keymap file. Do a few sanity checks
                     * to make sure it is a keymap, and not some bogus
                     * load file
                     */
                    if (TypeOfMem(kn->kn_Node.ln_Name)
                    &&  Stricmp(name,kn->kn_Node.ln_Name))
                    {
                        /* add to the system's keymap list */
                        AddHead(&(kr->kr_List),(struct Node *)kn);
                    }
                    else
                    {
                        /* bogus load file! Get rid of it and fail */
                        UnLoadSeg(segment);
                        kn = NULL;
                    }
                }
                else
                {
                    /* the keymap was added to the list behind our back!
                     * Free what was loaded and return happily
                     */
                    UnLoadSeg(segment);
                }
            }
        }

        /* get out of forbidden state */
        Permit();
    }

    if (kn)
        return(&kn->kn_KeyMap);

    return(NULL);
}
```