Libraries

COLLABORATORS

	TITLE : Libraries		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Libraries

1.1 Amiga® RKM Libraries: 13 Preferences

```
To make the Amiga operating system easily configurable by the user, the OS
comes with a family of editors and associated data files known
collectively as Preferences.  Preferences allows the user to set
system-wide configuration options such as the printer driver to use,
serial port baud rate and other items.  To make an application appealing
to the user, the system-wide Preferences settings should be respected as
much as possible by applications.  This chapter describes how to use the
Preferences system in your programs.

In Release 2 the number of Preference items and the way they are handled
is very different from 1.3 and earlier versions, though there is backward
compatibility with old Preferences items.  This chapter describes both the
old 1.3 and the new Release 2 Preferences.
```

1.2 13 Preferences / Preferences in 1.3 and Older Versions of the OS

```
In 1.3, the Preferences program allows the user to see and change many
system wide parameters, like the Workbench colors, pointer image, printer
settings etc.  When a Preferences item is changed, the new setting can be
used temporarily (until a reset occurs) or stored permanently in the
devs:system-configuration file.   Under 1.3, all Preferences items are
stored in this file which the system reads at boot time to find out how to
set various system-wide options.

The 1.3 Preferences system allows the user to change the following items:

   * Date and time of day. These are automatically saved in the
     battery-backed clock, if one is present.

   * Key repeat speed - the speed at which a key repeats when held down.
```

* Key repeat delay - the amount of delay before a key begins repeating.

* Mouse speed - how fast the pointer moves when the user moves the mouse.

* Double-click delay - maximum time allowed between the two clicks of a mouse double click. For information about how to test for double-click timeout, see the description of the DoubleClick() function in the Amiga ROM Kernel Reference Manual: Includes and Autodocs.

* Text size - size of the default font characters. The user can choose 64-column mode (64 characters on a line in high-resolution and 32 characters in low-resolution mode) or 80 column mode (80 characters on a line in high-resolution and 40 characters in low-resolution mode). The first variable in the Preferences structure is FontHeight, which is the height of the characters in display lines. If this is equal to the constant TOPAZ_EIGHTY, the user has chosen the 80-column mode. If it is equal to TOPAZ_SIXTY, the user has chosen the 64-column mode. Note that certain utility programs allow the user to change the default font under 1.3, so you cannot rely on the default font being Topaz 8 or 9.

* Display centering - allows the user to center the image on the video display.

* Serial port - the user can change the baud rate and other serial port parameters to accommodate whatever device is attached to the serial connector. Normally you use these values as defaults when you open the serial device. If you change the baud rate or other serial port options locally, it is good practice to reset them to the values specified in Preferences before quitting.

* Workbench colors - the user can change any of the four colors in the 1.3 Workbench screen by adjusting the amounts of red, green and blue in each color.

* Printer - the user can select from a number of printers supported by the Amiga and also indicate whether the printer is connected to the serial connector or the parallel connector.

* Print characteristics - the user can select paper size, right and left margin, continuous feed or single sheets, draft or letter quality, pitch and line spacing. For graphic printing, the user can specify the density, scaling method, select a vertical or horizontal dump, etc.

1.3 13 / Preferences in 1.3 and Older Versions / Reading 1.3 Preferences

Applications can obtain a copy of Preferences by calling the Intuition function GetPrefs(). In a system in which there is no devs:system-configuration file, GetDefPrefs() can be used to obtain the

Intuition default Preference settings.

```
    struct Preferences *GetPrefs(struct Preferences *preferences,
                                 LONG size);
    struct Preferences *GetDefPrefs(struct Preferences *preferences,
                                    LONG size);
```

GetPrefs() and GetDefPrefs() have two arguments, a pointer to a buffer to receive the copy of the user Preferences and the size of this buffer. The most commonly used data is grouped near the beginning of the Preferences structure and you are free to read only as much as you need. So, if you are only interested in the first part of the Preferences structure, you do not need to allocate a buffer large enough to hold the entire structure. These functions return a pointer to your buffer if successful, NULL otherwise.

If you are using Intuition IDCMP for input, you can set the IDCMP flag IDCMP_NEWPREFS (formerly the NEWPREFS flag under V34 and earlier versions of the OS). With this flag set, your program will receive an IntuiMessage informing you changes have been made to Preferences. To get the latest settings, you would again call GetPrefs().

1.4 13 / Preferences 1.3 & Older Versions / Preferences Structure in 1.3

The Preferences structure in 1.3 and earlier versions of the OS is a static 232 byte data structure defined in <intuition/preferences.h> as follows:

```
struct Preferences
{
    /* the default font height */
    BYTE FontHeight;                 /* height for system default font  */

    /* constant describing what's hooked up to the port */
    UBYTE PrinterPort;               /* printer port connection         */

    /* the baud rate of the port */
    UWORD BaudRate;                  /* baud rate for the serial port   */

    /* various timing rates */
    struct timeval KeyRptSpeed;      /* repeat speed for keyboard       */
    struct timeval KeyRptDelay;      /* Delay before keys repeat        */
    struct timeval DoubleClick;      /* Interval allowed between clicks */

    /* Intuition Pointer data */
    UWORD PointerMatrix[POINTERSIZE]; /* Definition of pointer sprite    */
    BYTE XOffset;                    /* X-Offset for active 'bit'       */
    BYTE YOffset;                    /* Y-Offset for active 'bit'       */
    UWORD color17;                   /**********************************/
    UWORD color18;                   /* Colours for sprite pointer      */
    UWORD color19;                   /**********************************/
    UWORD PointerTicks;              /* Sensitivity of the pointer      */

    /* Workbench Screen colors */
    UWORD color0;                    /**********************************/
    UWORD color1;                    /*  Standard default colours      */
    UWORD color2;                    /*   Used in the Workbench        */
    UWORD color3;                    /**********************************/

    /* positioning data for the Intuition View */
    BYTE ViewXOffset;                /* Offset for top lefthand corner  */
    BYTE ViewYOffset;                /* X and Y dimensions              */
    WORD ViewInitX, ViewInitY;       /* View initial offset values      */

    BOOL EnableCLI;                /* CLI availability switch (OBSOLETE)*/

    /* printer configurations */
    UWORD PrinterType;                   /* printer type                */
    UBYTE PrinterFilename[FILENAME_SIZE]; /* file for printer           */

    /* print format and quality configurations */
    UWORD PrintPitch;                /* print pitch                     */
    UWORD PrintQuality;              /* print quality                   */
    UWORD PrintSpacing;              /* number of lines per inch        */
    UWORD PrintLeftMargin;           /* left margin in characters       */
    UWORD PrintRightMargin;          /* right margin in characters      */
    UWORD PrintImage;                /* positive or negative            */
    UWORD PrintAspect;               /* horizontal or vertical          */
    UWORD PrintShade;                /* b&w, half-tone, or color        */
    WORD PrintThreshold;             /* darkness ctrl for b/w dumps     */

    /* print paper descriptors */
    UWORD PaperSize;                 /* paper size                      */
    UWORD PaperLength;               /* paper length in number of lines */
    UWORD PaperType;                 /* continuous or single sheet      */

    /* Serial device settings: These are 6 nibble-fields in 3 bytes     */
    /* (these look a little strange so the defaults will map out to 0)  */
    UBYTE   SerRWBits;   /* upper nibble = (8-number of read bits)      */
                         /* lower nibble = (8-number of write bits)     */
    UBYTE   SerStopBuf;  /* upper nibble = (number of stop bits - 1)    */
                         /* lower nibble = (table value for BufSize)    */
    UBYTE   SerParShk;   /* upper nibble = (value for Parity setting)   */
                         /* lower nibble = (value for Handshake mode)   */
    UBYTE   LaceWB;      /* if workbench is to be interlaced            */

    UBYTE   WorkName[FILENAME_SIZE]; /* temp file for printer           */

    BYTE    RowSizeChange;          /* affect NormalDisplayRows/Columns */
    BYTE    ColumnSizeChange;

    UWORD    PrintFlags;    /* user preference flags                    */
    UWORD    PrintMaxWidth; /* max width of printed picture in 10ths/in  */
    UWORD    PrintMaxHeight;/* max height of printed picture in 10ths/in */
    UBYTE    PrintDensity;  /* print density                            */
    UBYTE    PrintXOffset;  /* offset of printed picture in 10ths/inch   */

    UWORD    wb_Width;      /* override default workbench width  */
    UWORD    wb_Height;     /* override default workbench height */
    UBYTE    wb_Depth;      /* override default workbench depth  */
    UBYTE    ext_size;        /* extension information -- do not touch! */
                              /* extension size in blocks of 64 bytes   */
};
```

1.5 13 / Preferences in 1.3 and Older Versions / Setting 1.3 Preferences

The instance of the Preferences structure in memory can be changed with the Intuition SetPrefs() function:

```
    struct Preferences *SetPrefs(struct Preferences *preferences,
                                 LONG size, BOOL inform);
```

In addition to a buffer holding the Preferences structure, and the buffer size, this function takes an argument which indicates whether an IDCMP_NEWPREFS message should be broadcast to windows which have this flag set in the Window.IDCMPFlags field of their window.

```
    Avoid Using SetPrefs().
    -----------------------
    This function is normally only used by Preferences-like utilities.
    There should be no need for a normal application to set the system
    Preferences with SetPrefs().
```

1.6 13 / Preferences in 1.3 and Older Versions / Alternatives to SetPrefs

Since the Amiga is a multitasking system, it is rarely correct for a single Amiga application to modify the user's system-wide Preferences. Instead, use methods such as the following to modify only your own application's appearance or behavior.

* Custom screen applications can control their own display mode, resolution, palette, and fonts. Use functions such a LoadRGB4() to change your own screen's palette, and SetFont() to change your own screen and window fonts. Workbench applications should never change the attributes of the user's Workbench.

* The mouse pointer for a window may be changed with SetPointer().

* Serial device settings can be changed with the command SDCMD_SETPARAMS.

* Printer device settings may be changed by altering the printer's copy of the Preferences structure when you have the printer open. Note that Amiga applications should only keep the printer open while they are printing. This allows other applications to print, and also allows user changes to Printer Preferences to take effect.

See the Inutition and graphics chapters of this manual, and the "Printer Device" and "Serial Device" chapters of the Amiga ROM Kernel Reference Manual: Devices for more information.

1.7 13 Preferences / Preferences in Release 2

Under Release 2 (V36), the way Preferences are handled is significantly different. No longer is there one Preferences program with one configuration file. Instead there can be any number of Preferences editors (there are currently 13), each with its own separate configuration file covering a specific area. All these Preferences editors have the same look and feel. Using separate Preferences editors and configuration files allows for adding new Preferences items (and editors) in future versions of the OS.

1.8 13 / Preferences in Release 2 / Preferences Editors and Storage

In Release 2, the devs:system-configuration file has been replaced by various .prefs files, located in the ENV:sys and ENVARC:sys directories. System Preferences options currently in use are located in ENV:sys. Permanent, saved copies of system Preferences files are stored in ENVARC:sys. The contents of ENVARC: is copied at boot time to ENV:. Applications may also store their own preference files in ENV: but should use a subdirectory for that purpose.

Currently the following Preferences editors and files are available:

Table 13-1: Preferences Editors in Release 2

Preferences Editor	Preferences Configuration File
IControl	icontrol.prefs
Input	input.prefs
Palette	palette.ilbm
Pointer	pointer.ilbm
Printer	printer.prefs
PrinterGfx	printergfx.prefs
Overscan	overscan.prefs
ScreenMode	screenmode.prefs
Serial	serial.prefs
---	wbconfig.prefs
Font	wbfont.prefs, sysfont.prefs and screenfont.prefs
Time	---
WBPattern	wb.pat and win.pat

Each .prefs file is managed by editor with the same name, except for wbconfig.prefs, which is written directly by Workbench and has no editor. One Preferences editor has no .prefs file, Time. That Preferences editor writes directly to the battery backed clock.

When the user makes a change to a Preferences item with one of the editors, the changes will be saved in either ENV:sys or both ENV:sys and ENVARC:sys depending on whether the user saves the changes with the "Use" gadget or "Save" gadget of the Preferences editor.

The "Use" gadget is for making temporary changes and the new preferences will be stored only in ENV:sys. If the user reboots, the old preferences will be restored from the permanent copy in ENVARC:sys.

The "Save" gadget is for making permanent changes and the new preferences will be stored in both ENV:sys and ENVARC:sys. That way, if the user reboots, the new preferences will still be in effect since the system looks in ENVARC:sys to find out what preferences should be set to at boot time.

1.9 13 / Preferences in Release 2 / The ENV: Directory and Notification

One advantage of the new Preferences system in Release 2 is file notification. File notification is a form of interprocess communication available in Release 2 that allows an application to be automatically notified if a change is made to a specific file or directory. This makes it easy for the application to react to changes the user makes to Preferences files.

File notification is also used by the system itself. The Release 2 Preferences control program, IPrefs, sets up notification on most of the Preferences files in ENV:sys. If the user alters a Preferences item (normally this is done with a Preferences editor), the system will notify IPrefs about the change and IPrefs will attempt to alter the user’s environment to reflect the change.

For example, if the user opens the ScreenMode Preferences editor and changes the Workbench screen mode to high-resolution, the new settings are saved in Screenmode.prefs in the ENV:sys directory. IPrefs sets up notification on this file at boot time, so the file system will notify IPrefs of the change. IPrefs will read in the Screenmode.prefs file and reset the Workbench screen to high resolution mode.

Here’s a short example showing how to set up notification on the serial.prefs file in ENV:sys. The program displays a message in a window whenever this file is changed (e.g., when the user selects the "Use" or "Save" gadget in the Serial Preferences editor).

prefnotify.c

1.10 13 / Preferences in Release 2 / Preference File Format in Release 2

To understand the format of Preferences files, you must be familiar with IFF file standard (see the Amiga ROM Kernel Reference Manual: Devices for the complete specification).

In general all Preferences files are stored in the IFF format with a type of PREF (see the exceptions noted below). Each file contains at least two Chunks, a header Chunk and a data Chunk.

The Header Chunk   The Data Chunk

1.11 13 / / Preference File Format in Release 2 / The Header Chunk

The PRHD header chunk, contains a PrefHeader structure:

```
    struct PrefHeader
    {
        UBYTE ph_Version;
        UBYTE ph_Type;
        ULONG ph_Flags;
    };
```

Currently all the fields are set to NULL. In future revisions these fields may be used to indicate a particular version and contents of a PREF chunk.

1.12 13 / / Preference File Format in Release 2 / The Data Chunk

The data Chunk that follows the header Chunk depends on the kind of Preferences data the file contains. The types of Preferences data Chunks that are currently part of the system are:

Table 13-2: IFF Chunk Types in Release 2 Preferences Data Files

Chunk Name	Used With
FONT	Fonts, used for all font Preferences files. In future the PrefHeader may indicate what the font is used for.
ICTL	IControl
INPT	Input
OSCN	Overscan
PGFX	PrinterGfx
PTXT	PrinterText
SCRM	ScreenMode
SERL	Serial

Each chunk contains a structure applicable to the type.

1.13 13 / Preferences in Release 2 / FONT

```
    struct FontPrefs
    {
        LONG             fp_Reserved[4];
        UBYTE            fp_FrontPen;     /* Textcolor */
        UBYTE            fp_BackPen;      /* Character background color */
        UBYTE            fp_DrawMode;
        struct TextAttr  fp_TextAttr;
        BYTE             fp_Name[FONTNAMESIZE]; /* Font name */
    };
```

1.14 13 / Preferences in Release 2 / ICTL

```
    struct IControlPrefs
    {
        LONG  ic_Reserved[4];         /* System reserved            */
        UWORD ic_TimeOut;             /* Verify timeout             */
        WORD  ic_MetaDrag;            /* Meta drag mouse event      */
        ULONG ic_Flags;               /* IControl flags (see below) */
        UBYTE ic_WBtoFront;           /* CKey: WB to front          */
        UBYTE ic_FrontToBack;         /* CKey: front screen to back */
        UBYTE ic_ReqTrue;             /* CKey: Requester TRUE       */
        UBYTE ic_ReqFalse;            /* CKey: Requester FALSE      */
    };
```

```
The ic_Flags field can have the following values:

ICF_COERCE_COLORS
    This indicates that a displaymode with a matching number of colors
    has preference over a correct aspect ration when screen coercing
    takes place.

ICF_COERCE_LACE
    This indicates that chosing an interlaced display mode is allowed
    when coercing screens. Otherwise a non-interlaced display mode will
    be selected.

ICF_STRGAD_FILTER
    This indicates that control characters should be filtered out of
    string gadget user input.

ICF_MENUSNAP
    This indicates that an autoscroll screen should be snapped back to
    origin when the mouse menu-button is selected.

Note that the command key values in the last four fields of the
IControlPrefs structure are ANSI codes, not RAWKEY codes.
```

1.15 13 / Preferences in Release 2 / INPT

```
struct InputPrefs
{
    LONG            ip_Reserved[4];
    UWORD           ip_PointerTicks; /* Sensitivity of the pointer */
    struct timeval ip_DoubleClick;  /* Interval between clicks */
    struct timeval ip_KeyRptDelay;  /* keyboard repeat delay   */
    struct timeval ip_KeyRptSpeed;  /* Keyboard repeat speed   */
    WORD            ip_MouseAccel;   /* Mouse acceleration      */
};
```

1.16 13 / Preferences in Release 2 / OSCN

```
struct OverscanPrefs
{
    ULONG            os_Reserved[4];
    ULONG            os_DisplayID;   /* Displaymode ID */
    Point            os_ViewPos;     /* View X/Y Offset */
    Point            os_Text;        /* TEXT overscan dimension */
    struct Rectangle os_Standard;    /* STANDARD overscan dimension */
};
```

1.17 13 / Preferences in Release 2 / PGFX

```
struct PrinterGfxPrefs
{
    LONG  pg_Reserved[4];
    UWORD pg_Aspect;          /* Horizontal or vertical */
    UWORD pg_Shade;           /* B&W, Greyscale, Color */
    UWORD pg_Image;           /* Positive or negative image */
    WORD  pg_Threshold;       /* Black threshold */
    UBYTE pg_ColorCorrect;    /* RGB color correction */
    UBYTE pg_Dimensions;      /* Dimension type */
    UBYTE pg_Dithering;       /* Type of dithering */
    UWORD pg_GraphicFlags;    /* Rastport dump flags */
    UBYTE pg_PrintDensity;    /* Print density 1 - 7 */
    UWORD pg_PrintMaxWidth;   /* Maximum width */
    UWORD pg_PrintMaxHeight;  /* Maximum height */
    UBYTE pg_PrintXOffset;    /* X Offset */
    UBYTE pg_PrintYOffset;    /* Y Offset */
};
```

```
The possible values of each field are defined in <prefs/printergfx.h>.
Note that your application is responsible for checking if the supplied
values are valid.
```

1.18 13 / Preferences in Release 2 / PTXT

```
struct PrinterTxtPrefs
{
    LONG  pt_Reserved[4];            /* System reserved            */
    UBYTE pt_Driver[DRIVERNAMESIZE]; /* printer driver filename    */
    UBYTE pt_Port;                   /* printer port connection    */

    UWORD pt_PaperType;              /* Fanfold or single */
    UWORD pt_PaperSize;              /* Standard, Legal, A4, A3 etc. */
    UWORD pt_PaperLength;            /* Paper length in # of lines */

    UWORD pt_Pitch;                  /* Pica or Elite */
    UWORD pt_Spacing;                /* 6 or 8 LPI */
    UWORD pt_LeftMargin;             /* Left margin */
    UWORD pt_RightMargin;            /* Right margin */
    UWORD pt_Quality;                /* Draft or Letter */
};
```

1.19 13 / Preferences in Release 2 / SCRM

```
    struct ScreenModePrefs
    {
        ULONG sm_Reserved[4];
        ULONG sm_DisplayID;              /* Displaymode ID */
        UWORD sm_Width;                  /* Screen width */
        UWORD sm_Height;                 /* Screen height */
        UWORD sm_Depth;                  /* Screen depth */
        UWORD sm_Control;                /* BIT 0, Autoscroll yes/no */
    };
```

1.20 13 / Preferences in Release 2 / SERL

```
    struct SerialPrefs
    {
        LONG  sp_Reserved[4];     /* System reserved                */
        ULONG sp_BaudRate;        /* Baud rate                      */

        ULONG sp_InputBuffer;     /* Input buffer: 0 - 64K        */
        ULONG sp_OutputBuffer;    /* Future: Output: 0 - 64K, def 0 */

        UBYTE sp_InputHandshake;  /* Input handshaking              */
        UBYTE sp_OutputHandshake; /* Future: Output handshaking     */

        UBYTE sp_Parity;          /* Parity                         */
        UBYTE sp_BitsPerChar;     /* I/O bits per character         */
        UBYTE sp_StopBits;        /* Stop bits                      */
    };
```

1.21 13 / Preferences in Release 2 / Other File Formats in Release 2

Not every Preferences file is stored as an IFF file of type PREF. The palette.ilbm and pointer.ilbm files contain a regular ILBM FORM to store their imagery. The win.pat and wb.pat files use a raw format with 16 bytes reserved, followed by a WORD giving the total size of the pattern, a WORD giving the bitplane count, and byte arrays (currently 32 bytes) for each bitplane. The format of the wbconfig.prefs file is private.

1.22 13 Preferences / Function Reference

The following are brief descriptions of the system functions that relate to the use of Preferences. See the Amiga ROM Kernel Reference Manual: Includes and Autodocs for details on each function call.

Table 13-3: Functions Used with Preferences

Function	Description
GetPrefs()	Old 1.3 (V34) function for making a copy of the Preferences structure
SetPrefs()	Old 1.3 (V34) function for overwriting Preferences with new data
GetDefPrefs()	Old 1.3 (V34) function for copying default Preferences from ROM
StartNotify()	Release 2 DOS library function for monitoring a .prefs file for changes
EndNotify()	Ends notification started with StartNotify()
AllocIFF()	IFFParse library function that creates an IFFHandle for parsing
InitIFFasDOS()	Initialize the IFFHandle as a DOS stream
OpenIFF()	Initialize an IFFHandle for reading or writing a new stream
PropChunk()	Specify a property chunk to store
ParseIFF()	Parse an IFF file from the IFFHandle stream
CurrentChunk()	Returns the top level context of an IFF stream
FindProp()	Search for a property chunk previously declared with PropChunk()
CloseIFF()	Closes an IFF context opened with OpenIFF()
FreeIFF()	Frees the IFFHandle created with AllocIFF()