Harmony Release Message

contents:

Input/Output

*** Advance Warning: Changes to OPENFILE; multiple streams per file**

At some point in the future, the Interlisp-D i/o system will change so that each call to OPENFILE returns a distinct stream. This differs from the current behavior, inherited from Interlisp-10, that there can only be one stream open on any file, and that a second OPENFILE (assuming both are for INPUT) will return the same "opening". This change is required in order to deal rationally with files in a multiprocessing environment.

This change will of necessity produce the following incompatibilities:

1) OPENFILE will return a STREAM, not a full file name. To make this less confusing, STREAMs will have a print format that reveals the underlying

file’s actual name, and the functions UNPACKFILENAME and FILENAMEFIELD, when given a STREAM, will operate on the stream’s name.

2) A greater penalty will ensue for passing as the FILE argument to i/o operations anything other than the object returned from OPENFILE. Passing the file’s name will be significantly slower than passing the stream (even when passing the "full" file name), and in the case where there is more than one stream open on the file it might even act on the wrong one.

Advice for planning for this change:

Users are encouraged to write code which binds a variable to the result of OPENFILE and passes that variable to all i/o operations; such code will likely continue to work. Similar code that will work less well, if at all, is that which binds a variable to the result of an INFILEP and passes that to OPENFILE and all i/o operations; such code works well now, but implicitly assumes that INFILEP and OPENFILE return the same thing, an invalid assumption in this future world. (Code that passes incomplete file names to i/o operations is incurring a substantial performance penalty even now, and should have been changed long ago to use the result of the OPENFILE.)

To see more directly the effects of passing around STREAMs instead of file names, replace your calls to OPENFILE with calls to OPENSTREAM. OPENSTREAM is called in exactly the same way, but returns a STREAM. Streams can be passed to READ, PRINT, CLOSEF, etc just as the file’s full name can be currently, but using them is more efficient. The function FULLNAME, when applied to a stream, returns its full file name.

*** New function OPENSTRINGSTREAM: access strings like files** (243)

Interlisp-D inherited a feature from Interlisp-10 such that if a string was given as the file argument to an input function (READ, READC, etc.), that the function would interpret the string as the contents of a file and read the characters of the string. However, this never was a very clean design, and it interferes with the desire to use strings as file names. The following function has been created to handle i/o operations from/to strings more cleanly:

(OPENSTRINGSTREAM STR ACCESS)
Returns a stream that can be used to access the characters of the string STR. ACCESS may be either INPUT, OUTPUT, or BOTH; NIL defaults to INPUT. The stream returned may be used exactly like a file opened with the same access, except that output operations may not extend past the end of the original string. Also, string streams do not appear in the value of (OPENP).

*** Incompatible Change: GETFILEPTR, SETFILEPTR do not work with strings in the Harmony release**

In previous releases, the functions GETFILEPTR and SETFILEPTR could be used with strings. As of the Harmony release, this feature has been disabled, and calling these functions on strings will cause an error. The only way to use GETFILEPTR or SETFILEPTR with a string is to create a string stream with OPENSTRINGSTREAM.

*** Advance Warning: (READ <string>) will no longer read string characters in future release**

In the current release, (READ <string>) continues to work as before. However, in some future release, this feature will be decommissioned, and OPENSTRINGSTREAM will be the ONLY way to treat a string as a file. Users who depend on the old feature are encouraged to change their code now.

*** New function COPYCHARS for copying with EOL convention** (151 1967)

Many parts of the system have been changed to automatically convert between different EOL conventions. COPYFILE, MAKEFILE, and Tedit have been so modified, but we can't claim that every possible case has been taken care of. For user programs, the following function is available to do this conversion automatically:

(COPYCHARS SRCFIL DSTFIL START END)
This is like COPYBYTES, except that it performs the proper conversion if the EOL conventions of SRCFIL and DSTFIL are not the same. START and END are interpreted as byte specifications in SRCFIL. The number of bytes actually output to DSTFIL might be more or less than the number of bytes specified by START and END, depending on what the EOL conventions are. In the case where the EOL conventions happen to be the same, COPYCHARS simply calls COPYBYTES

*** 1100/1108 Parallel Port functions** (2503)

The 1100 has a parallel port connector with 8 bidirectional data lines, 8 unidirectional output lines, and 5 unidirectional input lines. The 1108 with Extended Processor Option (CPE) has a similar parallel port connector: the differences are (1) it has 6 unidirectional input lines vs. 5, (2) the power lines of the connector are 5 volts vs. 12, and (3) the pin layouts are different. A cable adapter is available to map the 1108's parallel port's pin layout into that of the 1100, or vice versa. The Interlisp functions WRITEPRINTERPORT and READPRINTERPORT are available for accessing the parallel port. For more information, see Appendix C (1100 & 1108 CPE Parallel Port).

*** "The infamous PEEKC bug" has been fixed: can backspace over PEEKC-ed chars** (826)

This was a longstanding bug in the Interlisp-D keyboard reader such that following (PEEKC T), the user couldn't backspace over the character that was peeked. This affected a number of functions, such as ASKUSER, FILES? and COMPILE, which peek at the first character of the user's typein.

*** Directory enumeration faster with multiple properties.** (113)

The directory enumeration code has been redone so that remote files do not have to be looked-up repeatedly when accessing multiple properties of files (size, author, etc). This improves the performance of DIR and the file browser.

*** DIRECTORY pattern interpretation improved** (1021 1087)

DIRECTORY and FILDIR have been modified to provide a consistent meaning for omitted fields in the file name pattern. Unspecified fields in the pattern default to *, except when the preceding field delimiter is included, in which case the field is explicitly null. Null version is interpreted as "highest". Thus:

```
DIR * = DIR *.* = DIR *.*;*
      enumerates everything.
DIR *. = DIR *.;*
      enumerates all versions of files with null extension.
DIR *.;
      enumerates highest version of files with null extension.
DIR *.*;
      enumerates highest version of everything.
```

Note: Some hosts/devices are not capable of supporting "highest version" in enumeration. Such hosts instead enumerate ALL versions.

*** WHENCLOSE operations called when streams are closed** (2045 514 1185)

Previously, the WHENCLOSE operations of a file, if any, were only invoked if CLOSEF was called with the file's name, not if called with the stream.

*** COPYFILE now uses FTP protocol whenever possible** (1302)

Previously, COPYFILE used the Leaf protocol when copying from a file server that also implemented Leaf. Using the FTP protocol for such file transfers is much more efficient for some servers.

*** COPYFILE infers file type when source file has none** (561)

COPYFILE always tries to create the new file with the same file type as the original file. If the original file's file type is unknown, COPYFILE now infers the type (file is BINARY if any of its 8-bit bytes have their high bit on). Previously, COPYFILE used the value of DEFAULTFILETYPE (initially TEXT), which was often the wrong thing to do, for example, when copying DCOM files.

*** COPYFILE to protected directory succeeds after password given** (458)

Previously, COPYFILE to a protected directory caused a "FILE WON'T OPEN" error even after asking for and receiving the correct password. This problem occurred only for "sequential" files written to a PupFtp server, not for ordinary MAKEFILEs.

*** COPYFILE to {CORE} or {DSK} copies file creation date** (1144 1145)

Previously, when copying a file to {CORE} or {DSK}, COPYFILE would ignore the old file's creation date and instead assign the current date and time as the new file's creation date. (COPYFILE to a remote file server has always copied the creation date correctly.)

*** Defining a user interrupt char no longer turns off previously defined user interrupts** (2068)

*** INTERRUPTCHAR user interrupts always "soft", no longer do CLEARBUF and FLASHWINDOW** (615)

In Interlisp-D, user interrupts set with INTERRUPTCHAR are always "soft", but are also "immediate", i.e., executing the interrupt does not disturb the process that is running or unwind the stack, but will happen at the next (interruptable) moment. Interrupts no longer clear the input buffer and flash the screen; users that want that behavior should explicitly call CLEARBUF and FLASHWINDOW as appropriate.

*** Control-C no longer calls RAID**

Control-C no longer calls RAID. For users who like to be able to use this low-level interrupt (more useful on 1100 than on 1108), it can be reenabled by executing (INTERRUPTCHAR 3 'RAID).

*** Shift-BS *NOT* equivalent to control-W** (2011)

The key Shift-BS has been changed so it is no longer equivalent to control-W in the initial Interlisp loadup. Users can change this by the appropriate call to KEYACTION, e.g. (KEYACTION 'BS '((8 23 NOLOCKSHIFT))) will restore the previous behavior.

*** UNPACKFILENAME works with strange file names: A.B.C** (144)

This is useful when accessing file servers which do not conform to Interlisp's file name conventions, such as NS file servers and UNIX-based file servers.

*** OPENFILE <Unix Leaf Server File> does not return filename with version ;0** (2281)

In some versions of the Unix leaf server code, for files without version numbers OPENFILE returns a filename with version zero. If this file name is then passed to OPENFILE again, it fails. Now, Interlisp explicitely looks for that situation, and strips off the version number entirely.

*** Known Bug: Unix FTP server returns file names like "FOO;1."** (2384)

Some versions of the Unix Ftp Server have a bug that causes DIR to print the names of extensionless files as, say, "FOO;1." (with a period AFTER the version number).

*** 1100/1132 {DSK} device supports file types** (629)

Files created by Interlisp on the 1100/1132 local file system now have TYPE information saved, where TYPE = TEXT or BINARY. Files written outside of Interlisp have TYPE = NIL.

*** Expanded documentation for RENAMEFILE** (2264)

(RENAMEFILE OLDFILE NEWFILE)
Renames OLDFILE to be NEWFILE. Causes an error, FILE NOT FOUND if OLDFILE does not exist. Returns the full name of the new file, if successful, else NIL if the rename cannot be performed. In the general case (e.g., when OLDFILE and NEWFILE are on different devices), RENAMEFILE works by copying OLDFILE to NEWFILE and then deleting OLDFILE.

*** Documentation correction: METASHIFT arg has to be T** (227)

The reference manual is incorrect when it says that the FLG argument to METASHIFT can be any non-NIL value. To work correctly, FLG must be T: (METASHIFT T). Other non-NIL values are passed as the ACTIONS argument to KEYACTION. The reason for this is that if someone has set Blank-bottom to some random behavior, then (RESETFORM (METASHIFT T) --) will correctly restore that random behavior.

*** (CLOSEF <display-stream>) is a no-op** (192)

*** GETECHOMODE checks its argument type** (2017)

GETECHOMODE will now generate an "ILLEGAL TERMINAL TABLE" error if it is passed an argument that is not a legal terminal table. Previously, it would not check its argument, and cause a more serious error if it was not a terminal table.

*** GETRAISE causes error if given bad terminal table argument, instead of calling RAID** (751)

Printing
========

*** Hardcopy functions cleaned up, documented**

In previous releases, the functions and variables used to send files to various printers have been redesigned repeatedly. We have been trying to design a simple interface that would "do the right thing" for most users, but would also allow users to get around the defaults when necessary. It was also important to provide facilities so users could define their own printers, and hook them into the normal hardcopy functions.

In the Harmony release, the hardcopy facilities have been simplified considerably. Files and bitmaps can be sent to the printer using the functions SEND.FILE.TO.PRINTER and HARDCOPYW. The variable DEFAULTPRINTINGHOST contains information about the available printers, and the variables PRINTERTYPES and PRINTFILETYPES contain the the information

necessary to print a file on any given printer. For full documentation, see Appendix A (Hardcopy Facilities).

*** Image streams allow printing arbitrary text and graphics on Press or Interpress printers** (2291)

Previously, the only documented way of printing text and graphics on Press or Interpress printers was to use one of the supported tools, such as Tedit. While these tools are sufficient for many needs, there was a need for functions that users could call from their programs to print arbitrary text and graphics. As part of a long-range effort to provide a simple, device-independent interface to the various graphics display routines, "image streams" were created.

An image stream is an output stream which "knows" how to process graphic commands. It can be passed as the FILE/STREAM argument to the ordinary character-output functions (PRINT, etc.) and to the graphic functions as well (DSPXPOSITION, DRAWCIRCLE, etc.). Some image streams, such as display and local-printer streams, may simply execute the appropriate operations to cause the desired image to appear immediately on the output medium. Other image streams (PRESS, INTERPRESS, etc.) interpret the graphic commands by saving information in a file of the appropriate format. If this file is on the {LPT} device, it will automatically be transmitted to a printer device when it is closed by CLOSEF. Non-LPT files can be transmitted later by explicit calls to LISTFILES and SEND.FILE.TO.PRINTER.

Image streams are created by the following function:

(OPENIMAGESTREAM FILE IMAGETYPE OPTIONS)
Opens and returns an output stream of type IMAGETYPE on a destination specified by FILE. IMAGETYPE can currently be either PRESS, INTERPRESS, or DISPLAY. Eventually, other image types will be implemented for other devices. FILE can name a file either on a normal storage device or on a printer device. In the latter case, the file is sent to the printer when the stream is closed.

FILE = NIL is equivalent to FILE = {LPT}. Names for printer files are of the form {LPT}PRINTERNAME.TYPE, where PRINTERNAME, TYPE, or both may be omitted. PRINTERNAME is the name of the particular printer to which the file will be transmitted on closing; it defaults to the first printer on DEFAULTPRINTINGHOST that can print IMAGETYPE files. The TYPE extension supplies the IMAGETYPE when it is defaulted (see below). OPENIMAGESTREAM will generate an error if the specified printer does not accept the kind of file specified by IMAGETYPE.

If IMAGETYPE is NIL, the image type is inferred from the extension field of FILE and the EXTENSIONS properties in the list PRINTFILETYPES. Thus, a PRESS extension denotes a Press-format stream, while IP, IPR, and INTERPRESS indicate Interpress format. If FILE is a printer file with no extension (of the form {LPT}PRINTERNAME), then IMAGETYPE will be the type that the indicated printer can print. If FILE has no extension but is not on the printer device {LPT}, then IMAGETYPE will default to the type accepted by the first printer on DEFAULTPRINTINGHOST.

Example: Assuming that IP: is an Interpress printer, P is a Press printer, and DEFAULTPRINTINGHOST is (IP: P):

(OPENIMAGESTREAM)
      returns an Interpress image stream on printer IP:

(OPENIMAGESTREAM NIL 'PRESS)
      returns a Press stream on P

(OPENIMAGESTREAM '{LPT}.INTERPRESS)
      returns an Interpress stream on IP:

(OPENIMAGESTREAM '{CORE}FOO.PRESS)
      returns a Press stream on the file {CORE}FOO.PRESS

If IMAGETYPE is DISPLAY, then the user is prompted for a window to open. The file name in this case will be used as the title of the window.

OPTIONS is a list in property list format that may be used to specify certain attributes of the image stream; not all attributes are meaningful or interpreted by all types of streams. Among the properties are:

REGION -- value is the region on the page (in stream scale units, 0,0 being the lower-left corner of the page) that text will fill up. It establishes the initial values for DSPLEFTMARGIN, DSPRIGHTMARGIN, DSPBOTTOMMARGIN (the point at which carriage returns cause page advancement) and DSPTOPMARGIN (where the stream is positioned at the beginning of a new page).

FONTS -- value is a list of fonts that are expected to be used in the stream. Some streams (e.g. Interpress) are more efficient if the expected fonts are called out in advance, but this is not necessary. The first font in this list will be the initial font of the stream, otherwise the DEFAULTFONT for that image type will be used.

HEADING -- the heading to be placed automatically on each page, NIL means no heading.

Other functions that are part of the device-independent graphics interface:

(IMAGESTREAMP X IMAGETYPE)
Returns X (possibly coerced to a stream) if it is an output image stream of type IMAGETYPE (or of any type if IMAGETYPE=NIL), otherwise NIL.

(IMAGESTREAMTYPE STREAM)
Returns the image type of STREAM.

(DSPSCALE SCALE STREAM)
Returns the scale of the image stream STREAM, a number indicating how many units in the streams coordinate system correspond to one screen point. For example, DSPSCALE returns 1 for display streams, and 35.27778 for Press and Interpress streams (the number of micas per screen point). In order to be device-independent, user graphics programs must either not specify position values absolutely, or must multiply absolute screen-point quantities by the DSPSCALE of the destination stream. (The SCALE argument to DSPSCALE is currently ignored; in future releases it will enable the scale of the stream to be changed under user control, so that the necessary multiplication will be done internal to the stream interface).

Note: Not all graphics operations can be properly executed for all image types. Currently, only display streams support BITBLT, FILLCIRCLE, and the dashing argument to DRAWCURVE. This functionality is still being developed, but even in the long run some operations may be beyond the physical or logical capabilities of some devices or image file formats. In these cases, the stream will approximate the specified image as best it can.

*** Can preview hardcopy on display using MAKEHARDCOPYSTREAM**

The fonts used in the printers are not exactly the same as the display fonts, because low-resolution screen fonts don't look good when printed on high-resolution printers. In particular, the character widths are not the same (even when scaled to take account of the printer resolution). Because of this, it is difficult to format text on the display so that it is EXACTLY where you want it, since it will be slightly different when printed. In order to allow users to "preview" hardcopy without actually printing it, the following functions are useful:

(MAKEHARDCOPYSTREAM DISPLAYSTREAM IMAGETYPE)

Changes the display stream so that measurements of character widths are consistent with the hardcopy device IMAGETYPE (PRESS, INTERPRESS, etc.). This is useful for seeing on the screen how an image will look when it is hardcopied. Caveat: This doesn't work for TEdit windows.

(UNMAKEHARDCOPYSTREAM DISPLAYSTREAM)
Changes a "hardcopy display stream" back into a regular display stream.

Note: When printing to a "hardcopy display stream", the text will not look as good as it will when printed. In particular, the characters may look crunched together. However, it accurately displays the relative positions of the letters, for formatting purposes.

*** Can print bitmaps on PRESS printers** (1206)

Bitmaps can be printed on press printers using HARDCOPYW, or by inserting bitmaps into Tedit documents. If the bitmap is too large for the press printer to handle (for example, if you try to print a complete screen image), it is clipped.

*** HARDCOPYW sends bitmaps to both PRESS and FULLPRESS printers** (1912)

HARDCOPYW now goes through FULLPRESSBITMAP when going to a full press printer. The function PRESSBITMAP uses the CLIPPINGREGION argument for clipping, while FULLPRESSBITMAP recognizes the SCALEFACTOR argument.

*** LISTFILES automatically detects and prints formatted Tedit files** (1147 225)

*** HARDCOPYW not a no-op if DEFAULTPRINTERTYPE=NIL** (1084)

*** Better error message printed if DEFAULTPRINTINGHOST is NIL** (1408)

Previously, attempting a hardcopy operation with DEFAULTPRINTINGHOST=NIL would produce an obscure low-level error.

*** HARDCOPYW has new arg: PRINTERTYPE** (456)

By default, HARDCOPYW will create an Interpress file if there are any Interpress printers on DEFAULTPRINTINGHOST. This default can be changed by passing PRESS as the PRINTERTYPE argument to HARDCOPYW.

*** HARDCOPY in the background menu does not reposition the cursor** (319)

Fonts

*** Incompatible Change: New font directory variables.** (155)

Previously, Interlisp used a confusing group of variables (FONTDIRECTORIES, NSFONTDIRECTORIES, NSFONTWIDTHSDIRECTORIES, STARFONTDIRECTORIES, FONTWIDTHSFILES) to determine where to search for font bitmap files and font widths files. These variables have been removed, and a new, rationally-named, set has been introduced:

DISPLAYFONTDIRECTORIES
Value is a list of directories searched to find font bitmap files for display fonts.

DISPLAYFONTEXTENSIONS
Value is a list of file extensions used when searching DISPLAYFONTDIRECTORIES for display fonts. Currently, Interlisp can read "STRIKE" and "AC" display font file formats. Eventually, all Interlisp display fonts will be distributed with the extension .DISPLAYFONT. Therefore, this variable should be initialized to (DISPLAYFONT STRIKE AC). Note that the extension on the file is used to locate the file, but once the

file is found Interlisp looks inside it to determine what format it is and how to read it. The function (FONTFILEFORMAT FILE LEAVEOPEN) returns the format of a font file (AC, STRIKE, etc.).

INTERPRESSFONTDIRECTORIES
Value is a list of directories searched to find font widths files for Interpress fonts. These files must have the extension "WD".

PRESSFONTWIDTHSFILES
Value is a list of files (not directories) searched to find font widths files for press fonts. Press font widths are packed into large "FONTS.WIDTHS" files.

All of these variables must be set before Interlisp can auto-load font files. These variables should be initialized in the site-specific INIT file.

*** Incompatible Change: FONTDESCRIPTOR datatype changed; Cannot load fonts dumped with UGLYVARS in Carol** (2183)

Between the Carol and Harmony releases, the system datatype FONTDESCRIPTOR was changed, to add a few more fields. Normally, changes to system datatypes do not affect users very much: they just have to recompile old files which use the datatypes. However, in the case that users saved Carol display fonts on files using the UGLYVARS file package command, more care is required to update these files to Harmony.

The problem is that datatype objects put on files with UGLYVARS contain the definition of the datatype. If a Carol file was loaded into Harmony which redefined a system datatype such as FONTDESCRIPTOR, Interlisp would almost certainly crash. In order to prevent people accidently redefining the FONTDESCRIPTOR datatype, the file package has been changed so that trying to change a datatype declaration while reading in an UGLYVARS object causes an error.

If users have created display fonts in Carol that they wish to use in Harmony, the upgrade procedure is the following: (1) While running the Carol release, load the lispusers package EDITFONT.DCOM. (2) Use the function (WRITESTRIKEFONTFILE <fontdescriptor> <filename>) to save each font descriptor as a "strike" format file. Note that strike files do not contain information about the font family, size, etc, so give the strike files descriptive names: e.g., GREEK10B.STRIKE. (3) While running the Harmony release, load EDITFONT.DCOM (note: the same package has been tested to work with both Carol and Harmony). (4) Use the function (READSTRIKEFONTFILE <family> <size> <face> <file>) to read in the strike file, and create a fontdescriptor with the specified family name, face, etc.

Note: It is recommended that user-created display fonts be stored as strike fonts, rather than stored as font descriptors on lisp files. If the files are named similarly to strike files distributed with Interlisp, and put on the same directories, they can be used like any other font.

*** Fontclasses are first-class data objects** (1416 1552)

Fontclasses have been introduced as a first-class data object which contains a set of related fonts for different devices. The font functions accept fontclasses, from which they extract the appropriate font for their device. The normal font class variables (DEFAULTFONT, CLISPFONT, etc.) are initialized to fontclass objects. Fontclasses are created and manipulated with the following functions:

(FONTCLASS NAME FONTLIST CREATEFORDEVICES)
Returns a new fontclass object with the name NAME and the device font components specified by FONTLIST, which should be a list of the form (<displayfont> <pressfont> <interpressfont> <otherfont1> <otherfont2> ...). <otherfontN> should be a list of the form (<devicename> <font>). Each of

the fonts in FONTLIST may be either a font descriptor, or a "font specification list" that FONTCREATE would accept. CREATEFORDEVICES is a list of the devices for which the fonts should be automatically created. Otherwise, the fonts are not actually created until they are accessed. Note: if a display font is specified in FONTLIST, it is always created.

(FONTCLASSCOMPONENT FONTCLASS DEVICE FONT NOERRORFLG)
Returns the font component of the fonctclass for the device DEVICE (DISPLAY, PRESS, INTERPRESS, etc.). If FONT is non-NIL, the specified component is replaced. If NOERRORFLG is non-NIL, FONTCLASSCOMPONENT return NIL if the component is unspecified in the fontclass, rather than causing an error.

Note: Because font classes are no longer represented by lists, old code which accesses the components of a font class with CAR, CADR, etc. will not work, and must be changed.

*** Font functions take font in many forms**

The font functions have been extended to take fonts specified in a variety of different ways. CHANGEFONT, DSPFONT, FONTCREATE, etc. can be applied to fontclasses, font descriptors, and "font lists" such as '(GACHA 10). The printout command ".FONT" has also been extended to accept fonts specified in any of these forms.

*** FONTCREATE accepts streams for DEVICE argument**

The function FONTCREATE has been extended so that the DEVICE argument can be an image stream, not just an image type. If a stream is given, the result will be a font appropriate for that stream.

*** New FONTPROP properties: SCALE, SPEC, DEVICESPEC**

The function FONTPROP has also been extended to recognize the new properties SCALE, SPEC, and DEVICESPEC. The value of the SCALE property is the units per screen-point in which the font is measured. For example, this is 35.27778 (the number of micas per screen point) for Press and Interpress fonts, which are measured in terms of micas. The value of the SPEC property is the full specification of the font as it is known to Interlisp, a family-size-face-rotation-device quintuple. The value of the DEVICESPEC property is the same as the value of the SPEC property, unless the system has had to coerce the font to another name to find the most appropriate rendering on a specific printing device.

*** New font function: FONTSAVAILABLE** (2274)

This function allows programs to determine what fonts are available for different devices.

(FONTSAVAILABLE FAMILY SIZE FACE ROTATION DEVICE CHECKFILESTOO?)
Returns a list of fonts that match the given specification. FAMILY, SIZE, FACE and DEVICE are the same as for FONTCREATE. Additionally, any of them can be the wildcard atom "*", in which case all values of that field are matched. In systems with several font directories, wildcard searches may take a while.

If CHECKFILESTOO? is NIL, only fonts already loaded into virtual memory will be considered. If CHECKFILESTOO? is non-NIL, the font directories for the specified device will be searched. When checking font files, the ROTATION is ignored.

Note: The search is conditional on the status of the server which holds the font. Thus a file server crash may prevent FONTCREATE from finding a file that an earlier FONTSAVAILABLE returned.

Each element of the list returned will be of the form (Family Size Face Rotation Device). For example:

(FONTSAVAILABLE 'MODERN 10 'MRR 0 'DISPLAY)

will return ((MODERN 10 (MEDIUM REGULAR REGULAR) 0 DISPLAY)) if Modern10 for the display is in virtual memory; NIL otherwise.

(FONTSAVAILABLE '* 14 '* '* 'INTERPRESS T)

will return a list of all the size 14 interpress fonts available either loaded into virtual memory or in the font directories.

*** SEE starts printing in correct font** (84)

Previously, if the default font of the exec window was changed, then SEE of an Interlisp source file would start out printing in that font (until the next font change). Now, SEE resets the font at the beginning of printing an Interlisp source file.

*** EDITCHAR, etc. now work with character # 256 (the dummy char)** (77)

1108 Local File System

*** Incompatible Change: Local file system format changed; MUST reformat 1108 disks**

The 1108 low-level disk format has been changed. To use the Harmony release, do the following: (1) copy any valuable local disk files to floppy or file server; (2) repartition the whole 1108 disk using the Harmony Installation Utility floppy (see the 1108 Users guide). Note that this erases ALL information on the disk; and (3) use DFSCREATEDIRECTORY to recreate any Lisp directories on local disk logical volumes.

Important Warning: Because of the change in disk format, you cannot run a Carol sysout on a Harmony-partitioned 1108, nor a Harmony sysout on a Carol-partitioned 1108. Attempting to do so may destroy information on the local disk.

*** Incompatible Change: Access logical volume FOO by {DSK}<FOO>, instead of {FOO}**

There is now a single local hard disk file device, {DSK}. Each logical volume with a Lisp directory on it now counts as a separate directory of the device {DSK}. (In Carol, each logical volume with a Lisp directory counted as a separate device.)

*** Incompatible Change: Many local file system functions renamed**

The user functions for the local file system have been redesigned. A number of functions have been renamed, and others have been added or deleted. The following functions have been renamed:

MKDIR ==> DFSCREATEDIRECTORY

MAKEPILOT ==> DFSPURGEDIRECTORY

DFSVOLUMES ==> VOLUMES

For more information, see the 1108 Users Guide.

*** (DISKPARTITION) returns the name of virtual memory logical volume** (390)

When Interlisp is running on an 1108, the function DISKPARTITION returns the name of the 1108 logical volume containing the currently-running Interlisp

virtual memory. This is analogous to the behavior of this function on 1100 or 1132s. The function CURRENTVOLUME has been removed.

*** SCAVENGEVOLUME now preserves filenames** (1789)

Note: SCAVENGEVOLUME is no longer included in the standard Interlisp-D system. It is available by loading the library package DlionFSScavenge.DCOM.

*** Running local file system functions on non-1108s will fail gracefully** (1049 1727 1761)

Previously, calling local file system functions from Interlisp running on a non-1108 would cause strange low-level errors. Now, all of these functions check whether Interlisp is running on an 1108, and generate an appropriate error message if not. One exception: VOLUMEDISPLAY simply returns NIL if not on an 1108, so this function can be called from init files that are run on different machines.

*** Local file system does not allocate large files all at once** (2372)

Previously, COPYFILE of a large file from an NS file server to the local file system would fail, because the local file system would try allocating the entire file first, and the connection would time out.

*** Local disk renames files from core device {DSK} to {PSEUDO-DSK}** (1663)

If a core device {DSK} exists when the local file system wants to create a device DSK for the local file system, a new device {PSEUDO-DSK} is created, any files are copied over, and the core device {DSK} is deleted. A warning message is also printed. If there are no files on the core device DSK, it is simply deleted with no warning message.

*** 1108 Local disk coerces HOST/DEVICE names to upper case** (808)

When returning full file names, the 1108 local file system coerces the "host/device" name to upper case: {dsk} -> {DSK}.

*** The VOLUMEDISPLAY window can be reshaped** (1044)

*** Known Bug: Local file system does not preserve file type information.** (2701)

1108 Floppy

*** Known bug: Should format new floppies before doing SYSOUT[{FLOPPY}]** (823)

There have been some cases where SYSOUT[{FLOPPY}] produced an incomplete sysout, when floppies that had never been formated before were used. Workaround: do (FLOPPY.FORMAT NIL NIL T) on new floppies first.

*** Floppy state flushed over LOGOUT** (1461)

Previously, if a floppy was left in a drive during LOGOUT and a new floppy was inserted before Interlisp was restarted, it was possible that Interlisp would use the old floppy directory information, which could destroy information on the new floppy. Now, floppy directory information is refetched the first time the floppy is used after Interlisp is restarted.

*** Trying to access FLOPPY on non-1108 no longer hangs** (1515)

Trying to access the floppy drive when running Interlisp on a machine other than the 1108 will now print "Floppy: No floppy drive on this machine" before generating the system error "FILE WON'T OPEN". Other operations,

such as DIR {FLOPPY}* will not generate an error, but instead make {FLOPPY} look like it has an empty directory so that file searches work correctly when {FLOPPY} is on the search path.

*** COPYFILE of SYSOUT from non-Interlisp floppies works correctly** (1965)

Previously, COPYFILE would not copy a sysout file from floppies correctly, if the sysout was not originally put there by Interlisp.

*** (OUTFILEP '{FLOPPY}xxx) works, instead of returning NIL** (1108)

*** New messages when creating multi-floppy file** (1579)

When copying a sysout or other large file to floppies (with FLOPPY.MODE = SYSOUT or HUGEPILOT), a message is printed at the start saying how many floppies will be required. Between floppies, the message now says "Insert Floppy #n", rather than "Insert next floppy".

*** (INFILEP '{FLOPPY}xxx) returns NIL if no floppy in the drive** (2255 2367)

Used to cause an error.

*** Floppy file versions on different floppy directories incremented correctly** (2389)

Previously, the floppy system ignored floppy file directories when computing the next version number for a new file. For example, it would create the file {FLOPPY}<BAR>NAME.;2 if there was a file {FLOPPY}<FOO>NAME.;1.

*** FLOPPY.MODE no longer changed to SYSOUT after a sysout** (207)

(SYSOUT '{FLOPPY}) will automatically change the floppy mode to SYSOUT during the sysout. However, after the sysout is completed, it is changed back to what it was before the sysout.

*** FLOPPY now supports file types** (628)

*** Floppy error msgs are printed in the typescript window, rather than the prompt window** (1495 1575)

*** (COPYFILE xx '{FLOPPY}) in PILOT mode gives error message** (996)

Previously, copying to {FLOPPY} giving a null file name could damage the information on the floppy. Now it just gives an error message.

*** Bad error msg "ARG NOT LP: NIL" changed** (217)

This obscure error message occurred when trying to read a Pilot floppy file when FLOPPY.MODE was set to SYSOUT. The appropriate work around was to execute (FLOPPY.MODE 'PILOT), and trying again.

*** FLOPPY.COMPACT accepts "No" as answer to confirmation** (1446)

*** Floppy errors are now regular file system errors** (8)

*** Sysout to write-protected floppy prints reasonable error message** (452)

RS232

*** 1108 optional RS232C port is supported; provides more reliable communication**

The Harmony release supports the optional RS232C port on the 1108. This port is buffered independently of Lisp operations, so there is little, if

any, chance of dropping characters. Use of this port requires the E30 hardware option.

*** RS232 documentation totally revised; new functions**

The RS232 documentation has been totally revised so it doesn't focus on the implementation on the Xerox 1100. A few of the minor additions explained in the new documentation:

The function RS232SHUTDOWN is a "cleaner" way of doing (CLOSEF '{RS232})

The function RS232INPUTSTRING inserts characters into the input ring buffer. This permits a way to simulate the reception of characters through the actual UART

The function RS232FORCEOUTPUT has a new argument which specifies whether the function should return before all the characters are transmitted (the default).

RS232DEVICEERRORFN is a new global variable used when handling hardware errors

The RS232LOGIN facility is now documented. This provides a way for automating the login procedure when connecting to various hosts.

*** Incompatible Change: Global variable RS232XON\XOFF? replaced by function RS232XON\XOFF?**

The interface to the XON/XOFF protocol has been changed: rather than setting the global variable RS232XON\XOFF?, the new function (RS232XON\XOFF? ON?) should be used to set and unset this state. In future release of the I/O processor code, enabling/disabling XON/XOFF processing by the RS232C port will require a functional interface to what is now merely a global variable.

*** RS232 no longer breaks over LOGOUT/restart** (1320 1393 2007)

*** RS232CHAT command ~LocalEcho works as specified** (1652)

NS File Servers

*** Many NS filing reliability problems fixed** (115 2010)

The NS filing system has been reworked, and made much more robust. In particular, a number of problems associated with open files timing out have been solved.

*** NS filing directory operations automatic** (218 219 86)

NS file servers support a true hierarchical file system, where subdirectories are just another kind of file. In previous releases of Interlisp-D, users had to explicitly create subdirectories using the function NSCREATEDIRECTORY. In Harmony, subdirectories are created automatically as needed: A call to OPENFILE to create a file in a non-existent subdirectory automatically creates the subdirectory; CONN to a non-existent subdirectory asks the user whether to create the directory. The function NSCREATEDIRECTORY has thus been removed. Note: Requires Services Release 8.0.

*** DIR fully enumerates NS files in subdirectories** (1504 2440)

In previous releases, DIR enumerated a directory only to the first level; it did not recursively enumerate the contents of subdirectories. In Harmony, DIR can enumerate a directory to arbitrary depth; the exact depth is

controlled by the variable FILING.ENUMERATION.DEPTH, which is a small positive integer or T. The default value is T, meaning infinite depth: the entire directory is enumerated, and subdirectory "files" do not appear at all. Also, the special function NSDIRECTORY is no longer needed, and has been removed: DIRECTORY works with NS file servers exactly as with other devices. Note: Requires Services Release 8.0. Earlier versions of Services will continue to behave as if FILING.ENUMERATION.DEPTH = 1.

*** NS file operations prompt for password** (605 722 2296 742)

When the user attempts an NS file server operation, Interlisp passes the current username and password (as given to the function LOGIN) to the NS file server. If these are not accepted, Interlisp prompts the user to enter the correct name and password. If the current username and password are correct, the user is not prompted at all.

Note: The user can abort an NS password prompt by typing control-E. The result of the file operation will be as if the NS file server did not exist.

*** SETFILEINFO, GETFILEINFO can access the TYPE attribute of NS files** (1708)

GETFILEINFO and SETFILEINFO now accept the TYPE and FILE.TYPE attributes for NS files. TYPE is the standard Lisp file type, with values TEXT and BINARY. FILE.TYPE is the (server-dependent) numeric value of the file's FILE.TYPE property, which a 16-bit number for NS file servers. Using the FILE.TYPE attribute, you can change the file type to other non-lisp file types.

*** GETFILEPTR works with NS files** (2058 2276)

SETFILEPTR of an NS file causes an error, since NS file servers do not currently support random access. However, GETFILEPTR now returns the correct character position for open files on NS file servers.

Note: SETFILEPTR works in the special case where the file is open for input, and the file pointer is being set forward. In this case, the intervening characters are automatically read.

*** (FULLNAME <file> 'NEW) and OUTFILEP now work for NS file servers.** (608)

Used to return NIL.

NS Print Servers

*** Can generate hardcopy of full screen on Xerox 8044 printer** (1163)

Use the HARDCOPYW function or the HARDCOPY command in the Background menu.

*** Multiple concurrent transmissions to NS printers now permitted** (240 1078)

Multiple concurrent NSPRINTs or HARDCOPYWs no longer confuse each other. No more breaks with "not an open NS socket".

*** ROTATATION argument to HARDCOPYW works to 8044 printers** (1616)

Previously, the rotation argument was not supported when sending bitmaps to the 8044 printer. Now, this is supported for ROTATION = a multiple of 90 degrees.

*** Can print to NS printers with A4 paper: variable NSPRINT.DEFAULT.MEDIUM** (2023)

The variable NSPRINT.DEFAULT.MEDIUM can be used to set the default NS printer medium. NIL (the default) means to use the printer's default; T means to use the first medium reported available by the printer; any other

value must be a Courier value of type MEDIUM. The format of this type is a list (PAPER (KNOWN.SIZE <TYPE>)) or (PAPER (OTHER.SIZE (<WIDTH> <LENGTH>))). The paper type <TYPE> is one of the atoms US.LETTER, US.LEGAL, A0 through A10, ISO.B0 through ISO.B10, and JIS.B0 through JIS.B10.

For European users who use A4 paper exclusively, it should be sufficient to set NSPRINT.DEFAULT.MEDIUM to (PAPER (KNOWN.SIZE "A4")).

Note: When using different paper sizes, it may be necessary to reset the variable DEFAULTPAGEREGION, the region on the page used for printing (measured in micas from the lower-left corner).

Ethernet Protocols

*** SPP, Courier, Clearinghouse reimplemented: low-level incompatibilities** (2440)

As part of the improvements to NS Filing and Printing, the underlying implementations of SPP, Courier and Clearinghouse have been substantially rewritten, in several places incompatibly. Users who program applications that use SPP, Courier or Clearinghouse directly should read Appendix D (NS Protocol Support).

*** "SPP Retransmit Queue out of order" errors fixed** (14 1664)

The SPP retransmit strategy has been completely revised, so this intermittent problem should disappear.

*** SETTIME now broadcasts for both PUP and NS time servers.** (1283)

SETTIME used to just try for a PUP time server.

*** Superfluous "not responding" messages after NS operations removed** (592 721)

Also, the function CLOSE.NSFILING.CONNECTIONS has been removed.

*** GETPUPSTRING applied to a blank pup now returns the null string instead of erroring** (772)

*** 1108s can "hear" their own Ethernet transmissions** (182)

The 1108 hardware is not capable of receiving the Ethernet packets it transmits. In previous releases, this meant that if an 1108 sent a packet addressed to itself, it would never receive it. In Harmony, the 1108 low-level Ethernet software takes care of this by faking receipt of such a packet. The implication of this is that programmers writing Lisp-based Ethernet servers can now test them out by running user and server code on the same machine.

*** Interlisp no longer hangs on an 1100 running 3MHz ethernet microcode without a 3MHz ethernet card** (485)

Window System

*** The ATTACHEDWINDOW package has been added to the standard Interlisp loadup.** (1767)

The ATTACHEDWINDOW library package has been added to the standard Interlisp-D window system. Many system tools (inspector, break package, etc.) now use attached windows for managing sets of windows. A number of changes have been made to the attached window facility:

New "attached prompt windows" provide a uniform way to access a small prompt window attached to another window.

Attached windows can be closed without closing the main window

BURY now buries all attached windows correctly

New function (DETACHALLWINDOWS MAINWINDOW) detaches and closes all windows attached to MAINWINDOW.

ATTACHMENU opens menu window immediately; new arg NOOPENFLG

New function MAINWINDOW for getting the mainwindow from an attached window

For complete documentation of the attached window facility, see Appendix B (Attached Windows).

*** Can move icons with LEFT button, expand with MIDDLE button** (1746 121)

Buttoning the LEFT button on an icon allows you to move it. Pressing the MIDDLE button expands it.

*** Changes to WFROMDS reduce empty tty windows** (1848)

WFROMDS has a new arg, DONTCREATE. If DONTCREATE is non-NIL, WFROMDS will never create a window, and return NIL if DISPLAYSTREAM does not have an associated window.

TTYDISPLAYSTREAM calls WFROMDS with DONTCREATE = T, so it will not create a window unnecessarily. Also, if WFROMDS does create a window, it calls CREATEW with NOOPENFLG = T. These changes fix many of the empty tty windows that used to appear.

*** Many changes to caret behavior** (544 1164)

There is now only one caret visible at any one time (except for TEdit which maintains its own caret). This fixes problems with carets being left on the screen and with windows being created just to take the caret down. The caret in the current process is always visible; if it is hidden by another window, its window is brought to the top. The function CARET has been changed, and the function CARETRATE, which changes the caret rate of the current process has been added:

(CARET NEWCARET)
Sets the shape that blinks at the location of the next output to the current process. NEWCARET is either (1) NIL - no changes, returns a CURSOR representing the current caret, (2) OFF - turns the caret off, (3) a CURSOR which gives the new caret shape or (4) T - resets the caret to the default which is the value of the variable DEFAULTCARET. DEFAULTCARET can be set to change the initial caret for new processes. The hotspot of NEWCARET indicates which point in the new caret bitmap should be located at the current output position. The previous caret is returned. Note: it is now permissible for the caret bitmap to be larger than cursor bitmap dimensions (16x16).

(CARETRATE ONRATE OFFRATE)
Sets the rate at which the caret for the current process will flash. The caret will be visible for ONRATE milliseconds, then not visible for OFFRATE milliseconds. If OFFRATE=NIL, the value of ONRATE is used. If ONRATE is T, both the "on" and "off" times are set to the value of the variable DEFAULTCARETRATE (initially 333). The previous value of CARETRATE is returned. If the caret is off, CARETRATE return NIL.

*** Caret flashing doesn’t bring window to top during buttoning or shift-selecting** (681)

The caret code has been changed so that it doesn't bring the flashing caret's window to the top if the user is buttoning or has a shift key down. This prevents the destination window (which has the tty and caret flashing) from interfering with the window one is trying to select text to copy from.

*** Cursor reset correctly after going through scroll bar** (378)

Previously, slowly dragging the mouse out of the left of a Tedit window would change the cursor to a right-facing arrow (in the left margin), change it to the scrolling cursor (in the scroll bar), and "restore" it to the right-facing arrow upon leaving the scoll bar. The window system now restores the cursor correctly to the value of DEFAULTCURSOR upon leaving a window.

*** New Background menu when Copy-key pressed; allows copy-inserting a SNAP** (1808)

Various system utilities (TEdit, DEdit, TTYIN) allow information to be "copy-inserted" at the current cursor position by selecting it with the "copy" key held down. (Normally the shift keys are the "copy" key, this action can be changed in the key action table.) It is now possible to "copy-insert" the bitmap of a snap into a Tedit document. If the right mouse button is pressed in the background with the copy key held down, a menu with the single item "SNAP" appears. If this item is selected, the user is prompted to select a region, and a bitmap containing the bits in that region of the screen is inserted into the current tty process, if that process is able to accept image objects (like Tedit).

This is implemented by the new variables BackgroundCopyMenu and BackgroundCopyMenuCommands, which are interpreted similar to BackgroundMenu and BackgroundMenuCommands. If the right mouse button is pressed in the background when the copy key is down, the menu stored in the variable BackgroundCopyMenu is envoked. If this is NIL, a new menu is created from the menu commands in BackgroundCopyMenuCommands.

*** RESHAPEBYREPAINTFN uses new strategy to determine window contents after reshape** (1613)

Previously, RESHAPEBYREPAINTFN (the default reshaping function) always copied the old image to the lower-left corner of the new window, adding any new image to the top and left. This produced unintuitive results in the case where the lower left corner was grabbed and moved out. The new behavior will display the part of the object in the direction of the expansion (if the opposite side is not moved) and only display white space beyond the extent if the extent is fully visible.

This change required that a fourth argument be passed to the RESHAPEFN of a window: OLDSCREENREGION, the region that the window occupied before being reshaped. This allows RESHAPEBYREPAINTFN to determine which edges of the window have been moved. Note: in some situations, RESHAPEBYREPAINTFN may call a window's REPAINTFN as many as four times on different window regions.

*** New Background Button Event Functions** (637 682)

The variables BACKGROUNDBUTTONEVENTFN, BACKGROUNDCURSORINFN, BACKGROUNDCURSOROUTFN and BACKGROUNDCURSORMOVEDFN provide a way of taking action when there is cursor action when the cursor is in the background. If set to the name of a function, that function will be called, respectively, whenever the cursor is in the background and a button changes, when the cursor moves into the background from a window, when the cursor moved out of the background into a window, and when the cursor moves from one place in the background to another. These are analogous to the window properties BUTTONEVENTFN, CURSORINFN, CURSOROUTFN, and CURSORMOVEDFN.

*** New BURYW behavior -- faster algorithm** (741)

BURYW has been changed to take down the windows overlapping the window to be buried, then reopening them in the right order.

*** 1108 background border preserved over LOGOUT/restart** (876 2277 106)

The backgound border (around the screen) on the 1108 can be changed with the function CHANGEBACKGROUNDBORDER. During LOGOUT, the border is changed back to the default shade. Now, the border is restored to its new pattern after LOGOUT/restart on an 1108.

*** Fixed: Caret didn't flash on 1108 after LOGOUT/restart** (511)

Sometimes, the caret would not flash after doing LOGOUT and restarting Interlisp on an 1108. This could be fixed by typing (CARET), so it was not a major problem, but it was annoying.

*** If MENU is called with RELEASECONTROLFLG=T, the menu window is brought to the top.** (241)

Previously, a "released" menu could be hidden by other windows. Now, the released menu will stay visible until it is closed or an item is selected.

*** Moving an off-screen window onto the screen redisplays its contents** (1945)

*** DRAWCURVE works correctly in INVERT mode if BRUSH=1** (1978)

DRAWCURVE, DRAWCIRCLE, and DRAWELLIPSE to the display will work if the brush argument is 1, and the "operation" of the displaystream is INVERT. For brushes larger than 1, these functions will still use the ERASE operation.

DRAWCURVE to other image streams generally only supports the PAINT operation.

*** New window property: NOSCROLLBARS** (1053)

If a window's NOSCROLLBARS property is non-NIL, scroll bars will not be brought up for the window, even if it has both EXTENT and SCROLLFN properties. This allows the creation of windows that can scroll ONLY under program control.

*** New window property: WINDOWTITLESHADE sets shade used in window title bar.** (1054 1148 1354)

If a window's WINDOWTITLESHADE property is non-NIL, it should be a texture which is used as the "backgound texture" for the title bar on the top of the window. If this property is NIL, then the value of the variable WINDOWTITLESHADE is used, initially black. Note that black is always used as the background of the title printed in the title bar, so that the letters can be read -- the remaining space is painted with the "title shade".

*** Textures can be BITMAPs up to 16 by 16 bits** (449)

TEXTUREP, BITBLT, DSPTEXTURE, DSPFILL, etc. accept bitmaps up to 16 bits wide by 16 bits high as textures. When a region is being filled with a bitmap texture, the texture is treated as if it were 16 bits wide (if less, the rest is filled with white space).

*** New functions INVERTW, FLASHWINDOW** (1153)

(INVERTW WIN SHADE)
Inverts the window WIN, by XOR-ing it with the shade SHADE. If SHADE=NIL, the default is to XOR with the shade BLACK, which simply inverts the bits.

(FLASHWINDOW WIN? N FLASHINTERVAL SHADE)

Flashes the window WIN?, by inverting it twice. N is the number of times to flash the window (default is once). FLASHINTERVAL is the number of milliseconds to wait with the window inverted (default is 200). SHADE is interpreted as in INVERTW.

If WIN? is NIL, the whole screen is flashed. In this case, the SHADE argument is ignored (can only invert the screen).

*** New function DECODE.WINDOW.ARG: coerces window specs to window** (775)

(DECODE.WINDOW.ARG WHERESPEC WIDTH HEIGHT TITLE BORDER NOOPENFLG)
This is a useful function for creating windows. WHERESPEC can be a WINDOW, a REGION, a POSITION, or NIL. If WHERESPEC is a WINDOW, it is returned. In all other cases, CREATEW is called with the arguments TITLE, BORDER, and NOOPENFLG. The REGION argument to CREATEW is determined from WHERESPEC as follows:

If WHERESPEC is a REGION, it is adjusted to be on the screen, then passed to CREATEW. If WIDTH and HEIGHT are not numbers, CREATEW is given NIL as a REGION argument.

If WIDTH and HEIGHT are numbers and WHERESPEC is a POSITION, the region whose lower left corner is WHERESPEC, whose width is WIDTH and whose height is HEIGHT is adjusted to be on the screen, then passed to CREATEW.

If WIDTH and HEIGHT are numbers and WHERESPEC is not a POSITION, then GETBOXREGION is called to prompt the user for the position of a region that is WIDTH by HEIGHT.

If WIDTH and HEIGHT are used, they are used as interior dimensions for the window.

*** New function MAKEWITHINREGION: moves region within another region** (775)

(MAKEWITHINREGION REGION LIMITREGION)
Changes (destructively modifies) the left and bottom of the region REGION so that it is within the region LIMITREGION, if possible. If the dimensions of REGION are larger than LIMITREGION, REGION is moved to the lower left of LIMITREGION. If LIMITREGION is NIL, the value of the variable WHOLEDISPLAY (the screen region) is used. MAKEWITHINREGION returns REGION.

*** INSIDEP now accepts a window as its REGION arg** (1151)

If the REGION arg to INSIDEP is a window, the window’s interior (its clipping region) is used.

*** REGIONP now true for regions whose components are floating point numbers.** (893)

Previously, only integers were allowed as components of a region.

*** EXPANDBITMAP works without the color package loaded** (674)

EXPANDBITMAP uses the function \FAST4BIT, which was previously only defined in the color library package. \FAST4BIT has been added to the standard Interlisp loadup.

*** READBITMAP, PRINTBITMAP have new argument: FILE** (538)

(READBITMAP FILE)
(PRINTBITMAP BITMAP FILE)
These functions can now be used to read and print bitmaps to arbitrary files, without changing the primary input/output stream.

*** CURSORINFN and CURSOROUTFN window properties extended** (242)

The CURSORINFN and CURSOROUTFN window properties can now be lists of functions as well as single functions. All functions on the list are called.

*** Scrollbar provides better indication when contents are above the window** (698)

Previously, there were some cases where the scrollbar would not hit the bottom unless the bottom of the extent was a small distance above the top of the window.

*** EDITBM does not reposition the cursor to the center of the screen** (769)

*** control-D during EDITSHADE now closes the window** (854)

*** (BITMAPHEIGHT <texture>) now gives an error for non-bitmap textures** (1315)

*** If CHANGEOFFSETFLG menu property is non-NIL, popup menus come up in correct position** (1641)

Previously, they would come up one pixel above and to the right of where they were last time (relative to the cursor).

*** Scrolling works correctly after changing window border size** (1763)

*** If LEFT button down, GETREGION calls NEWREGIONFN with MOVINGPOINT = NIL** (1578)

Previously, if GETREGION was called when one of the mouse buttons is already down (LEFT), the first call to NEWREGIONFN did not have MOVINGPOINT = NIL

*** EXPANDW no longer fails if called on expanded window** (1588)

*** (DSPCREATE <bad-arg>) signals "ILLEGAL ARG" error, instead of going into RAID** (302)

*** DRAWCURVE no longer generates an error if dashing is non-NIL** (1614)

*** ADDMENU/DELETEMENU do not modify the menu for subsequent use** (522)

Tedit

*** New Tedit page formatting facilities**

Tedit now includes facilities for specifying the page layout to be used when a document is formatted and printed. The user can now control page formatting such as page numbers, headings, multiple columns, etc.

*** Tedit has separate menus for Para looks, Char looks, and Page Looks** (581)

This solves a number of problems. In particular, it is no longer necessary to scroll a single long menu up and down to set and apply character and paragraph looks.

*** Control-E can be used to abort Get, Put, etc. commands** (642)

After selecting the Tedit commands Get, Put, Include, etc. from the title menu, the user is asked to type in a file name. The operation can be aborted at this time simply by typing control-E.

*** Can shrink an unsaved Tedit document** (1535)

Previously, Tedit caused an error, when it tried to print the file name of the document in the icon. Now, it detects this situation and creates an empty icon.

*** Tedit more careful about erasing caret images on the screen** (933)

*** EOFP works correctly for text streams** (1582)

*** Tedit hardcopy uses {DSK} to store large files, so larger files can be printed** (870)

*** Tedit uses more compact representation for bitmaps** (1801)

The format of bitmaps in Tedit files has been changed. This new format should take up about half the space, and it can be read/written many times as fast of the old format. It does not do any compression. The old bitmap-reading functions have not been removed, so old bitmaps will be converted as they are encountered.

Dedit

*** "BREAK" or "()" of top-level expression no longer causes stack overflow** (850 959)

*** Process switch from DEDIT to TEDIT won't cause it to ignore tabs** (636)

*** DEdit's internal data structures revised to take 1/3 less space** (678)

This should improve swapping performance over extended programming sessions.

*** (DF <undefined function>) creates blank function template.** (678 739 961)

If DF is called on a name with no function definition, the user is prompted with "No FNS defn for <function name>. Do you wish to edit a dummy defn?". If the user confirms (by clicking left-button), a "blank" definition is displayed in the Dedit window. If any changes are made, on exit from the editor, the definition will be installed as the name's function definition. Exiting the editor with the STOP command will prevent any changes to the function definition.

If DF is called with a second arg of NEW, as in (DF <function name> NEW), a blank definition will be edited whether the function already has a definition or not.

*** Inserting huge piece of code no longer causes bad screen extent.** (400)

Previously, after inserting a huge piece of code into a function, Dedit could lose track of the size of the function, so the user could not scroll up enough to see the last part of the inserted code.

*** DEDIT REPAINTFN redisplays selection highlighting** (254)

*** Comments print correctly when inserted or SWITCHed** (431)

*** Deleting first dotted-pair from list of pairs reprints correctly** (816)

*** Buttoning in the Dedit Edit Buffer switches the current process.** (188)

Previously, you had to button in the main Dedit window.

*** Dedit menu only comes to the top when Dedit is the TTY process.** (678)

*** "Shouldn't Happen! DEDITDSPS tangled" errors reduced** (849 995)

Previously, this could happen if you called the inspector from Dedit (by EVAL-ing (INSPECT ...)), and called Dedit from the inspector window. This particular symptom has been cured in the current Dedit. However, exiting DEdit processes out of order can still cause this error.

*** Dedit supports the COPY key on the 1108 keyboard** (228)

Shift select supports both the COPY and the right shift keys.

*** EditOps menu follows when the main Dedit window is moved** (359)

*** Edit buffer doesn't attach to incorrect window** (745)

Previously, after "TTYIn Form" of atom, the Dedit Typein window for DEDIT sometimes would attach itself to the bottom of random windows on the screen.

*** Dedit doesn't reprint function on exit.** (678)

Previously, in some situations Dedit would reprint the entire function after exit, as a side effect of changing the edit date comment.

*** !UNDO command is now undoable** (366)

*** Double deletes give better error message** (1877)

Previously, if one deleted a deleted selection (in a serious of commands with the control key down), Dedit would break with the error "Shouldnt: No MapEntry". Now, Dedit detects this situation, prints out the error message "Cant: Already deleted!", and doesn't cause a break.

*** CAP command capitalizes first letter of atom** (945)

It used to do the same as RAISE, capitalizing all the letters of the item selected.

*** "?=" command in Dedit works for fns of no arguments** (15)

Used to give "xxx not a function" error message.

Break Package

*** Editor called from display break package in broken process** (162 1263)

Inspecting a function in the display frame window now calls the editor in the broken process. Thus variables evaluated in the editor will be in the broken process.

*** Can now REVERT to any frame on the stack.** (512 583 2074 20 1349)

Previously, there were restrictions on reverting to internal "DUMMY" frames, because it could cause the system to crash or freeze. Now, REVERT has been fixed so that it is safe to revert to any frame on the stack.

*** Break windows are not opened on "STORAGE FULL" errors.** (1309)

This is similar to the treatment of "ARRAYS FULL" errors. In either case, allocating storage for a break window would cause the error to occur repeatedly.

*** Typing control-B in a break window no longer gives "Break within Break" error** (520)

*** AUTOBACKTRACEFLG extended: can cause BT for NON-error breaks** (734)

Previously, if AUTOBACKTRACEFLG was non-NIL, then the command BT would be executed automatically on error breaks, but not on user breaks (calls to functions broken by BREAK). It has been extended as follows: If AUTOBACKTRACEFLG is NIL (the default), no backtrace is brought up. If its value is T, then on error breaks the BT menu is brought up. If its value is BT!, then on error breaks the BT! menu is brought up. If its value is ALWAYS, then on any break the BT menu is brought up. If its value is ALWAYS!, then on any break the BT! menu is brought up.

*** ERRORTYPELST is now a SPECVAR** (11)

It makes sense for users to change the global value of ERRORTYPELST, but programs that rebind it clearly want changed behavior only in their own stack context. It is only looked up under error conditions, so it shouldn't cause a performance problem.

*** Break package more careful about aborting process on closing window** (162)

Closing a break window now only aborts the associated process if it was in tty wait and the closed window was the tty window. This should stop some inadvertant aborts.

*** Warning: Typed-in BT, BTV commands don't start at top of stack** (990)

When a stack frame name is selected in the backtrace menu, the variable LASTPOS is set to the selected stack frame. This allows breaks commands such as REVERT, ?=, etc. to use the selected frame. However, the value of LASTPOS also indicates to the break commands BT, BTV, BTV!, etc. where to start listing the stack.

Inspector

*** Using SET to set inspector values no longer creates many TTY windows** (31)

Previously, the inspector SET command would create a new window for the user to type a value every time it was used. Now, the default SET routine uses an attached prompt window on top of the inspect window to receive the new value.

*** New inspect window commands: "IT_datum", "IT_selection"** (142)

The values displayed in an inspect window can be accessed by commands on the menu brought up by pressing the MIDDLE button in the title of the window. The command "IT_datum" sets the variable IT to the object being inspected in this window. The command "IT_selection" sets the variable IT to the current property name or value selected in the inspect window.

*** Variable INSPECTPRINTLEVEL used for printing inspector values** (435)

When the inspector prints field values, PRINTLEVEL is reset to the value of INSPECTPRINTLEVEL, initially (2 . 5).

*** Inspector calls INSPECTCODE to inspect compiled code objects** (640)

CHAT

*** Chat does not turn off interrupt characters until AFTER creating the Chat window** (799)

This allows the user to abort the call to Chat by typing control-E while specifying the Chat window region.

*** Reshaping Chat window does not change terminal type** (844)

Previously, reshaping a Chat window caused Chat to reassert the terminal type specified when the connection was first opened. If the user in the meantime had told the remote host that the terminal type was different, then this would set it back.

*** Chat grabs TTY as soon as it starts to reconnect** (611)

Previously, the "reconnect" menu button didn't switch the tty to the chat process until the connection was reestablished.

*** Chat ignores the padding character DEL** (789)

*** Chat display no longer off by 1 character after EMACS insert operation** (349 1629 1261)

Newer versions of EMACS perform character insertion by an unusual sequence that Chat was not emulating correctly.

*** Chat in EMACS mode updates cursor position promptly** (1256)

Previously there was a bug that deferred the cursor update following a positioning command with the mouse until the next type-in occurred.

*** Chat displays the EMACS mode state in the window title** (1221)

When Chat EMACS-mode is on, "EMACS ON" is printed in the Chat window title.

TTYIN
=====

*** Incompatible change: EDITPREFIXCHAR default is NIL** (51)

The variable EDITPREFIXCHAR is now by default NIL, meaning there is initially no TTYIN prefix meta-character defined. This change was made to avoid confusing users who don't use TTYIN editing commands. If you want to be able to issue editing commands to TTYIN, you should either call (TTYINMETA T) to enable bottom-blank (STOP on 1108's) as a true meta key, or set EDITPREFIXCHAR to the character code of your preferred meta prefix (it used to be 193, for top-blank).

*** TTYIN is enabled in break windows created by control-B during type-in** (1929 1399 91)

*** FIX command with TTYIN prettyprints history events** (28)

The programmer's assistant command FIX calls TTYIN to edit the text of the history event. TTYIN now prettyprints the event for ease of editing.

*** Typing control-E under TTYIN won't cause "NON-NUMERIC ARG" error** (16)

*** Typein lines starting with ";" no longer erased** (1512)

Previously, TTYIN interpreted a line starting with the character ";" as a comment, and would ignore it, erasing the line from the screen. Although ";" is defined on LISPXHISTORYMACROS as a no-op anyway, TTYIN's behavior was inappropriate in cases where one was not typing to the Lisp exec.

The old behavior is still available for those desiring it: if the first character on a line of typein is equal to the variable TTYINCOMMENTCHAR (a

character code or NIL), then the line is erased, and no input function will see it. TTYINCOMMENTCHAR is initially NIL.

Stack & Interpreter

*** Known Bug: Must do (HARDRESET) after stack overflow, or else second stack overflow gives fatal error** (1927)

If a stack overflow occurs, rather than type "^" to escape from the break, do a hardreset. Otherwise, the NEXT stack overflow may cause an unrecoverable error. Either evaluate (HARDRESET) from the break window, or type control-D from Teleraid.

*** New function: EVALHOOK** (1168 1769 777)

(EVALHOOK FORM EVALHOOKFN)
EVALHOOK evaluates the expression FORM, and returns its value. While evaluating FORM, the function EVAL behaves in a special way. Whenever a list other than FORM itself is to be evaluated, whether implicitly or via an explicit call to EVAL, EVALHOOKFN is invoked (it should be a function), with the form to be evaluated as its argument. EVALHOOKFN is then responsible for evaluating the form; whatever is returned is assumed to be the result of evaluating the form. During the execution of EVALHOOKFN, this special evaluation is turned off. (Note that EVALHOOK does not effect the evaluations of variables, only of lists).

Here is an example of a simple tracing routine that uses the EVALHOOK feature:

```
_(DEFINEQ (PRINTHOOK (FORM)
      (printout T "eval: " FORM T)
      (EVALHOOK FORM (FUNCTION PRINTHOOK]
(PRINTHOOK)
_(EVALHOOK '(LIST (CONS 1 2) (CONS 3 4)) 'PRINTHOOK)
eval: (CONS 1 2)
eval: (CONS 3 4)
((1 . 2) (3 . 4))
```

*** Internal arithmetic functions changed to have the "right" frame name** (1807 1886)

In compiled code, a call to a primitive arithmetic function, such as PLUS, turns into a Lisp opcode, which normally executes entirely in microcode. In exceptional cases, however, the microcode executes a call on an internal arithmetic function, such as \SLOWPLUS2. Previously, if an error occurred in such a function, the backtrace contained the internal function name, rather than the name you would expect from looking at the source code. This has been changed so that the frame names of internal arithmetic functions are the appropriate user-level functions.

*** EVALV has new argument RELFLG: release-stack-ptr flag** (191)

Most of the stack evaluation functions (ENVEVAL, etc.) have a flag argument which determines whether the stack pointer will be automatically released. To be consistent, EVALV now has a RELFLG argument, even though it doesn't strictly need it (EVALV is guaranteed to return, unlike the other functions).

*** (APPLY*) now gives "UNDEFINED FUNCTION: NIL" error** (1678)

History and Exec

*** BREAK, TRACE, SEE, etc. recognize quoted arguments: new function NLAMBDA.ARGS** (1722 593)

A number of NLAMBDA functions now recognize if their argument is quoted. For example, (BREAK 'FOO) will now break the function FOO, rather than the function QUOTE. LISPX macros and commands which normally take their args unquoted (DIR, CONN, etc.) also work with quoted arguments. For example, typing DIR 'FOO* is now the same as DIR FOO*.

This change was accomplished by defining a new function (NLAMBDA.ARGS X). This interprets its argument as a list of unevaluated nlambda arguments. If any of the elements in this list are of the form (QUOTE ...), the enclosing QUOTE is stripped off. Actually, NLAMBDA.ARGS stops processing the list after the first non-quoted argument. Therefore, whereas (NLAMBDA.ARGS '((QUOTE FOO) BAR)) -> (FOO BAR), (NLAMBDA.ARGS '(FOO (QUOTE BAR))) -> (FOO (QUOTE BAR)).

*** Error correction of function name doesn't lose args** (337)

Previously, if one had an NLAMBDA nospread function FOO, one could type "FOO ALPHA" to the exec and FOO would be run, with ALPHA as its argument. If however, one mistyped FOO (as foo, FOOX,etc.) and the spelling corrector sucessfully corrected it to FOO, the exec did not pass the arguments along. This has been fixed.

*** PRINTLEVEL UNDO-able from top level exec** (141)

Typing PRINTLEVEL to the top-level exec will substitute a call to the undoable function /PRINTLEVEL.

File Package

Incompatible Change: Source/DCOM file location algorithm changed (100 671 509 1666)

Each Interlisp source and compiled code file contains the full filename of the file, including the host and directory names, in a FILECREATED expression. The compiled code file also contains the full file name of the source file it was created from. Previously, the file package used this information to locate the appropriate source file when "remaking" or recompiling a file.

This has turned out to be a bad feature in distributed environments, where users frequently move files from one place to another, or where files are stored on removable media. For example, suppose you MAKEFILE to a floppy, and then copy the file to a file server. If you load and edit the file from a file server, and try to do MAKEFILE, it will break, trying to locate the source file on a floppy, which is probably no longer loaded.

In the Harmony release, the file package searches for the source file on the connected directory, and on the directory search path (on the variable DIRECTORIES). If it is not found, the host/directory information from the FILECREATED expression be used.

Warning: One situation where the new algorithm does the wrong thing is if you explicitly LOADFROM a file that is not on your directory search path. Future MAKEFILEs and CLEANUPs will search the connected directory and DIRECTORIES to find the source file, rather than using the file that the LOADFROM was done from. Even if the correct file is on the directory search path, you could still create a bad file if there is another version of the file in an earlier directory on the search path. In general, you should either explicitly specify the SOURCEFILE argument to MAKEFILE to tell it

where to get the old source, or connect to the directory where the correct source file is.

*** HPRINT, UGLYVARS, HORRIBLEVARS don't redeclare datatypes** (2251)

The file package commands UGLYVARS and HORRIBLEVARS call the function HPRINT to print out loadable representations of arbitrary data structures. If a data structure contains an instance of an Interlisp datatype, the datatype declaration is also printed onto the file.

This has causes problems when a system datatype declaration dumped into a file doesn't match the current declaration. Redefining a system datatype will almost definitely crash Interlisp. The Interlisp system datatypes do not change very often, but there is always a possibility when loading in old files created under an old Interlisp release.

To prevent accidental system crashes, HREAD has been changed so that loading an HPRINTed structure will NOT redefine datatypes. Instead, it will cause an error "attempt to read DATATYPE with different field specification than currently defined". Continuing from this error will redefine the datatype.

*** Incompatible change: User INIT files are now loaded normally, and appear on FILELST** (638 122 1822)

Previously, the user init files were SYSLOAD-ed, and their filecoms were not saved. This was inconvenient when people wanted to modify their init files. Now, they are loaded with LDFLG=NIL, so their filecoms are saved, and they appear on FILELST. Note that the system "site" init file is still loaded with SYSLOAD.

The function GREET has been changed as follows:

The system greet file (GREETFILENAME T) is loaded with the SYSLOAD parameter. The user greet file (GREETFILENAME <username>) is loaded with normal file package settings, but also under errorset protection and with PRETTYHEADER set to NIL to suppress the "FILE CREATED" message.

Note: Users should try to make sure that their init file is "undoable". If they use the file package command "P" to put expressions on the file to be evaluated, they should use the "undoable" version, e.g. /SETSYNTAX rather than SETSYNTAX, etc. This is so another user can come up, do a (GREET) and have the first user's initialization undone.

*** MAKEFILE "remake" option asks whether to load DONTCOPY expressions** (1881 83 2312)

When a MAKEFILE is performed with the "remake" option to copy definitions from an old file, MAKEFILE checks to see if all of the necessary definitions had been loaded from the old file. In the past, if you had only loaded the compiled version of a file with (DECLARE: .. DONTCOPY ..) expressions, MAKEFILE would automatically and quietly load the definitions from the old file. In some circumstances this could be disastrous -- if the user had circumvented the file package in some way, and loading the old definitions overwrote new ones.

MAKEFILE now asks before performing these operations, e.g.

"Only the compiled version of FOO was loaded, do you want to LOADVARS the (DECLARE: .. DONTCOPY ..) expressions from {DSK}<MYDIR>FOO.;3?"

*** HASDEF with SOURCE=? calls WHEREIS database package if loaded** (735)

According to the documentation, passing SOURCE=? to the file package type functions should try (among other options) calling the function WHEREIS with FILES=T, which will search the WHEREIS hashfile database if the WHEREIS package is loaded. In the case of HASDEF called with SOURCE=?, WHEREIS was

being called with FILES=NIL, so the WHEREIS package was not being used. This produced strange behavior in Dedit, such that evaluating (DF <system-function>) would load and edit the function, but selecting the function in a Dedit window and buttoning "Edit" would not.

*** I.S.OPRS now works as a file package "type" for COPYDEF and UNSAVEDEF** (1734)

*** (* * X ...) no longer signifies that X is a filevar** (417 1620)

When a form such as (FNS * FOOFNS) appears in the filecoms of a file, this means that the list of functions should be taken from the variable FOOFNS. In this case, FOOFNS is known as a filevar.

Previously, there was a bug with comments of the form (* * this is a comment), where the first word of the comment ("this") was interpreted as a filevar. This had some strange consequences, such as the first words of such comments appeared in (FILECOMSLST xxx 'VARS), and these atoms were set to NOBIND if the file was loaded with LDFLG=SYSLOAD.

*** Comments allowed in file package commands** (1936)

The file package now allows comments to appear in most places in the filecoms. For example:
(INITVARS (* this is a comment) (FOO 5)).

*** Default setting of CLEANUPOPTIONS changed to (RC)** (1817)

Previously, the default value of CLEANUPOPTIONS was (LIST RC), so CLEANUP would list and recompile all files. If you wish to retain that behavior, simply reset CLEANUPOPTIONS.

*** (PF <function> <file>) prints message if file not found, or function not found on file** (1832)

Previously, PF just returned NIL if either the function was not found or the file was not found.

*** DC FOO can find file FOO.LSP** (271)

Previously, the user had to type DC FOO.LSP to edit the coms of a file with a non-NIL extension.

*** ADDTOFILE prompt changed from "new file?" to "create new file XXX?"** (1942 1234)

Compiler

*** Incompatible change: Default RECOMPILEDEFAULT changed from EXPRS to CHANGES** (1670 1786)

Previously, the default value of RECOMPILEDEFAULT was EXPRS. This meant that when recompiling a file, those functions currently defined by EXPRs would be recompiled. Generally, this is a good indication of which functions had been edited. However, a problem occurs if the user explicitly calls COMPILE to compile a particular function. A later RECOMPILE or CLEANUP would not recompile that function. By setting the default RECOMPILEDEFAULT to CHANGES, RECOMPILE or CLEANUP will recompile those functions which have been changed according to the FILECREATED expression in the source file. Under some circumstances, this may cause functions to be recompiled unnecessarily, but it is safer.

Benefits of RECOMPILEDEFAULT=CHANGES:

If you normally load a source file, edit a few functions, then MAKEFILE and RECOMPILE, the effect of the change to RECOMPILEDEFAULT is that fewer functions are recompiled (only the ones you changed, not all the functions on the file).

If you normally load the compiled file, then LOADFROM the source, and are running with DFNFLG = PROP, so that edited functions are not unsaved, then the effect of the change is that the edited functions do get recompiled, even though they are not defined by EXPRs.

Disadvantages of RECOMPILEDEFAULT=CHANGES:

If you go thru several rounds of the edit-makefile-recompile loop, then possibly MORE functions are recompiled than necessary, since each RECOMPILE will compile ALL the functions that have changed since you first LOADFROMed the file, not just the ones changed since the last recompile.

When Masterscope advises you to UNSAVEDEF a set of functions containing occurrences of records or macros that changed, the unsaving will have NO effect on which functions later get recompiled. You need to set RECOMPILEDEFAULT = EXPRS in order for this to work right.

*** Incompatible change: Compiling with mode=ST or STF redefines functions, even if DFNFLG=PROP** (1673)

Previously, when the compiler "redefined" a function, it respected the value of DFNFLG. If DFNFLG=PROP, the compiler put the new definition on the CODE property instead of in the definition cell of the function.

The new behavior is that as functions are compiled, they really ARE "stored and redefined"; the new compiled definition is placed in the definition cell, even though DFNFLG=PROP.

The new behavior is less confusing, but if you are used to the old behavior, be careful. If you run with DFNFLG=PROP to completely avoid inadvertantly redefining something in your running system, you MUST use compile mode F, not ST.

*** Warning: Compiler modified, so don’t load Harmony-compiled files into old sysouts.** (1570)

A number of modifications have been made to the compiler, which might cause backward incompatibility. In general, old compiled code will work in new releases of Interlisp-D, but compiling in a NEW release and loading into an OLD release is not guaranteed to work.

*** STore-and-Forget option to COMPILE no longer leaves EXPRs on property list** (423)

*** LOADTIMECONSTANT works in interpreted code** (800 1176)

*** Compiler prints warning if user code attempts to bind a variable previously declared as a constant** (277)

Masterscope

*** Masterscope CHECK command smarter about CONSTANTS, blocks** (303)

The CHECK command now knows about CONSTANTS. Previously, constants were treated like any other variable, and CHECK printed a warning if they were used freely without being declared. Also, CHECK now omits the preamble "in no block" (followed by a list of functions) when a file has no block declarations.

*** Show Paths browser properly updated when redisplayed** (1110)

When a function in a SHOW PATHS browser graph is edited, the window "greys out", to indicate that (possibly) some of the information has changed. Previously, under some circumstances, when a greyed out browser window was redisplayed, Masterscope would not reanalyze the functions that had changed.

*** ". SHOW WHERE X CALLS Y" now finds lowest (not highest) level macro containing call** (1878)

*** Masterscope HELP command removed** (1872)

This used to print out a two-page summary of the Masterscope commands, which was not very useful in finding out how to use Masterscope.

DWIM & CLISP

*** Advance Warning: In future releases, (CLISPDEC 'MIXED) will be default** (2032)

In past releases of Interlisp, and in the Harmony release, the default clisp declaration is FIXED, which means that all clisp constructs are translated using integer arithmetic, unless the user explicitly changes the declaration. Therefore, (A+B) translates into (IPLUS A B), and (for X from A to B do ...) is translated using integer arithmetic to increment X and compare it to B.

In Interlisp-D, mixed (generic) arithmetic is not appreciably slower than integer arithmetic, so we are trying to convert the system to use generic arithmetic as much as possible.

Therefore, starting with the next release, the default clisp declaration will be MIXED, so generic arithmetic functions will be used when translating clisp constructs. (A+B) will translate into (PLUS A B), and (for X from A to B do ...) will be translated using PLUS and GREATERP. Of course, the user can change this declaration using CLISPDEC.

We do not expect that this change will effect any programs: the only conceivable problems could be in constructs like (for X from A to B do ...) where the programmer COUNTED on floating A and B being converted to fixed point before the loop.

*** Macro-expansion now independent of DWIM** (1212)

Previously, macro-expansion was handled by the MACROTRAN entry on DWIMUSERFORMS. This meant that macros would only be interpreted if DWIM was turned on. The macro-expansion machinery has been moved to a much 'higher' level (closer to the source), before DWIMFLG is tested and a large amount of otherwise unnecessary processing was done. This means that macro expansion can continue even when users turn off DWIM.

*** New variable DWIMINMACROSFLG controls whether args to macros are dwimified** (1074)

If the variable DWIMINMACROSFLG = T (the default), DWIM will recursively dwimify the arguments to macros (i.e. macros will be treated like LAMBDA functions). If DWIMINMACROSFLG = NIL, arguments to macros are not dwimified.

To provide finer control over the interpretation of individual macros, DWIM uses the INFO property of the macro name: If the INFO prop is or contains the atom EVAL, the macro arguments are dwimified, even if

DWIMINMACROSFLG=NIL. If the INFO prop is or contains the atom NOEVAL, the macro arguments are not dwimified, even if DWIMINMACROSFLG = T.

*** DWIM no longer tries to interpret type-in as edit commands** (1211 1439)

Previously, one of the actions DWIM took on unbound atom or undefined function errors was: "if the atom is an edit command, envoke the editor on the last thing edited, passing the atom as an edit command". DWIM is of necessity 'heuristic', attempting to second guess what the user meant. However, this correction is one that, over time, has become wrong far more often than right.

*** Incompatible Change: DWIMIFYENGLISH, CLISPENG package totally de-supported** (1425)

The "feature" of translating English into Lisp documented in the 1978 Interlisp Reference Manual is no longer supported in Interlisp-D. The lispusers package CLISPENG is no longer supported, either.

*** DWIM tries upper-casing undefined functions and unbound atoms** (2136)

*** DWIM now gives warning on coercion from lower to upper case** (454 395)

Previously, DWIM would upper-case atoms and functions without warning or notification, which caused a great deal of confusion. Now, the default is to print a warning "=XX" when coercing from "xx" to "XX". This feature is controlled by the variable FIXSPELL.UPPERCASE.QUIET (initially NIL). If non-NIL, no warning is given.

*** CLISPIFY does not translate (fetch A.B of X) to X:A.B** (1057)

In the case where a record field has a period in it, it is inappropriate for CLISPIFY to translate a fetch or replace statement into the more concise form X:A.B, since DWIM interprets "A.B" as the "data path" rather than the field name.

*** RUNONFLG initialized to NIL in the default environment** (1669)

If the variable RUNONFLG = T, DWIM will attempt "run-on" spelling corrections, breaking up unknown names. From experience, it seems that this hurts more often then it helps. Therefore, the default has been changed so this feature is initially disabled.

*** FIXSPELL only moves words on "real" spelling lists** (867)

When spelling-correcting words on the system spelling lists SPELLINGS1, SPELLINGS2, etc, FIXSPELL moves words to the front of the list when a word is successfully corrected. However, this is not necessarily the correct behavior for user-supplied spelling lists, where it may be wrong to alter the order of the list. If FIXSPELL is called with DONTMOVETOPFLG = non-NIL, words are not moved in the spelling list. As an additional check, FIXSPELL won't move correct words to the front of a spelling list unless the spelling list contains the special marker used to separate the temporary and permanent sections of the system spelling lists (the value of SPELLSTR1).

*** I.S.OPRS work even if CLISPFLG=NIL** (1802)

Contrary to the documentation, some iterative statement operators would not be translated correctly is CLISPFLG was NIL, because their definition included forms such as $$VAL_T. These operators now work even if "_" is disabled, either specially or because CLISPFLG is NIL.

Performance Tools

==========

*** DOSTATS removed from standard Lisp loadup** (1768)

Since the SPY package provides most of the functionality of DOSTATS, in addition to being usable on Xerox 1108's, the function DOSTATS has been removed from the standard Interlisp system. The code for DOSTATS is available by loading in the library files PCALLSTATS and APS (automatically loaded by PCALLSTATS).

*** DOSTATS now resets DFNFLG and compiler optimizations** (802)

Previously, it was possible that DOSTATS would collect stats on the wrong form if DFNFLG was set improperly. For example, if DFNFLG=PROP, the form would be put on a property list, and stats would be collected for whatever happened to be in the definition cell of STATSDUMMYFUNCTION. Also, DOSTATS didn't reset compiler optimizations, so that it might "optimize" forms like (IQUOTIENT 1234567 -1) into a constant.

*** Control-D out of DOSTATS stops statistics-gathering** (124)

Previously, typing control-D during the execution of DOSTATS would stop the computation, but wouldn't stop the gathering of statistics. This was a serious problem, because very quickly the disk would fill up and Interlisp would fall into SWAT, losing everything. Now, exiting DOSTATS with control-D automatically turns off statistics-gathering.

*** TIMEALL now compiles form with optimizations ON** (1780)

If TIMEALL is called with #TIMES>1, a dummy form is created, compiled and executed #TIMES times, to provide more accurate measurement of small computations. Previously, this compilation was done with optimizations off if running multiple times. In the face of objections, this has been changed: now TIMEALL compiles the dummy form with compiler optimizations ON.

Warning: An important result of this change is that it is not meaningful to use TIMEALL with very simple forms that are optimized out by the compiler. For example, (TIMEALL '(IPLUS 2 3) 1000) will time a compiled function which simply returns the number 5, since (IPLUS 2 3) is optimized to the integer 5.

*** BREAKDOWN overhead reduced** (1353 1994)

The per-call overhead to BREAKDOWN has been substantially reduced, which should give much more meaningful results.

Storage & Data Types
==========

*** Incompatible Change: ARRAY default type is POINTER, FLOATP is stored unboxed** (1381 1061 1464)

If NIL is given as the TYPE argument to ARRAY, the default array type is POINTER, not DOUBLEPOINTER. Anyone using the DOUBLEPOINTER mechanism should change any instances of (ARRAY x) to (ARRAY x 'DOUBLEPOINTER).

Arrays of type FLOATP are now stored unboxed. This increases the space and time efficiency of FLOATP arrays. Users who want to use boxed floating point numbers should use an array of type POINTER instead of FLOATP.

*** Advance Warning: CAR or CDR of non-list will cause error in future releases; new variable CAR/CDRERR** (768 685)

According to the Interlisp Reference Manual, the value of applying the functions CAR and CDR to a non-list (other than NIL) is undefined. In Interlisp-D, the actual action depended on the data type: (CAR <atom>) returned NIL, (CDR <atom>) returned the atom's property list, (CAR <anything else>) returned the string "{car of non-list}", and (CDR <anything else>) returned the string "{cdr of non-list}".

This has turned out to be a bad design. This design typically caused obscure bugs in programs which CDR down a list, and stop on NIL. If the tail of the list is not NIL, then the program loops endlessly, taking CDR of "{cdr of non-list}". This problem also occurs with functions like (FMEMB A B), which loop endlessly if B is not a list.

Because of these problems, the Interlisp maintainers decided that CAR and CDR should cause an error on non-lists. Places in the system code which used the old conventions have been cleaned up. In future releases, the default will be changed so that CAR or CDR of non-NIL non-lists will cause errors. This will also effect system functions, such as FMEMB, which use CAR and CDR. User programs which depend on the old conventions will have to be modified.

To root out functions in the system which rely on the old CAR/CDR convention, the global variable CAR/CDRERR has been created.

If CAR/CDRRERR=NIL (the current default), then CAR and CDR act as they always have, returning a string for non-lists. If CAR/CDRERR=T, then CAR and CDR of a non-list (other than NIL) causes an error.

If CAR/CDRRERR=ONCE, then CAR and CDR of a string causes an error, but CAR/CDR of anything else returns the string "{c...r of non-list}" as before. This catches loops which repeatedly take CAR or CDR of an object, but it allows one-time errors to pass undetected.

If CAR/CDRERR=CDR, then CAR of a non-list returns "{car of non-list}" as before, but CDR of a non-list causes an error. This setting is based on the observation that nearly all infinite loops involving non-lists occur from taking CDRs, but a fair amount of careless code takes CAR of something it has not tested to be a list

*** MKATOM no longer loops forever when the atom hash table is full** (866)

Previously, running out of atoms (the limit is currently ~32K) would cause an infinite loop. Now, Interlisp will cause a storage full error when there are about 7 "pages" of atom space left, and will call RAID (MP 9323 on an 1108) when there are no more atoms left.

*** Hash arrays have been totally reimplemented; better performance, interface** (1096)

The hash array facility has been totally reimplemented, to improve performance and provide a better interface to the overflow behavior. Old programs using hash arrays will still work, but not as efficiently as if they were recoded to take advantage of the new implementation.

In the old implementation, the hash array functions accepted a list whose CAR was a hash array datum. If the hash array overflowed during some hash array operation, the action taken (error, automatically enlarging the hash array, etc.) was determined by the CDR of the hash array list.

In the new implementation, the "overflow method" is stored as part of the hash array datatype. The hashing functions will operate correctly on "old-style" hash arrays of the form (harrayp . overflow), but more slowly than with "new-style" hash arrays that contain their overflow methods.

New functions:

(HASHARRAY MINKEYS OVERFLOW)
Creates a hash array containing at least MINKEYS hash keys, with overflow method OVERFLOW (if NIL, the default overflow method is to expand the size of the hasharray and rehash all the entries). The function HARRAY still exists for backward compatibility, equivalent to (HASHARRAY MINKEYS 'ERROR).

(HARRAYPROP HARRAY PROP NEWVALUE)
Returns the property PROP of HARRAY; PROP can have the system-defined values SIZE (returns the maximum occupancy of HARRAY), NUMKEYS (number of occupied slots), or OVERFLOW (overflow method). In the case of OVERFLOW, a new method may be specified as NEWVALUE.

(HASHARRAYP X)
Returns X if X is either an old- or new-style hash array (i.e a hash array datum or a list whose car is a hash array datum). Otherwise returns NIL.

(HARRAYP X)
Returns X if it is a hash array datum, as returned by the function HARRAY or HASHARRAY. Unlike HASHARRAYP, this returns NIL for lists whose CAR is a hash array datum. HASHARRAYP should probably be used instead in most circumstances.

*** STORAGE changes: new arguments; prints free list info** (63 1815)

The function STORAGE in Interlisp-D now takes two optional arguments for filtering the amount of information presented:

(STORAGE TYPES PAGETHRESHOLD)

If TYPES is given, STORAGE only lists statistics for the specified types. TYPES is an atom or list of types. If PAGETHRESHOLD is given, then STORAGE only lists statistics for types that have at least PAGETHRESHOLD pages allocated to them.

Note: These optional arguments are different from the optional arguments to STORAGE in Interlisp-10.

STORAGE now prints out more information about the size of the entries on the array free list, including a breakdown of the free block sizes. The block sizes are broken down by the value of the variable STORAGE.ARRAYSIZES, initially (4 10 100 1000 4000 NIL), which yields a printout of the form:

```
 variable-datum free list:
le 4            11 items;       44 cells.
le 10           34 items;      240 cells.
le 100          39 items;     1619 cells.
le 1000         25 items;     7856 cells.
le 4000          2 items;     2449 cells.
others           0 items;        0 cells.
```

This information can be useful in determining if the variable-length data space is fragmented. If most of the free space is composed of small items, then the allocator may not be able to find room for large items, and will extend the variable datum space. If this is extended too much, this could cause an ARRAYS FULL error, even if there is lots of space left in little chunks. This information is primarily of use to system programmers.

*** New CASEARRAY arg for STRPOS** (900)

STRPOS has been extended to take a new argument CASEARRAY. If non-NIL, this should be a casearray like that given to FILEPOS. The casearray is used to map the string characters before comparing them to the search string. See the documentation for FILEPOS, CASEARRAY, etc. in the reference manual.

*** New BACKWARDSFLG arg for STRPOS, STRPOSL** (900)

If non-NIL, this argument specifies that the search should be done backwards from the end of the string.

*** Incompatible Change: LDIFFERENCE always returns copy of list: resolves Interlisp-D/10 difference** (318)

Previously, if (LDIFFERENCE FOO BAR) was EQUAL to FOO (ie, FOO and BAR shared no elements), Interlisp-D would return a result which is EQ to FOO, while Interlisp-10 would return a copy of FOO. Interlisp-D has been changed to make it compatible with the Interlisp-10 behavior.

*** Interpreted REPLACE of a data with a BITS field now correct.** (1502)

Previously, the interpreted version of REPLACEFIELD would do the wrong thing if called to replace a datatype declared with a BITS field. This only affected interpreted calls to REPLACE and not compiled calls.

*** (CREATE ... SMASHING ...) translates into more efficient form** (1343)

The translation of (CREATE ... SMASHING ...) forms has been changed for RECORD and TYPERECORD records, to produce forms that execute more efficiently when compiled.

*** The atoms NIL and T now can have property lists** (915 916 924)

*** (APPEND '(A . B)) now runs correctly when compiled** (1411)

Previously, (APPEND '(A . B)) returned (A . B) when interpreted, (A) compiled. Now, it returns (A . B) always.

*** ELT, SETA error changed from "ILLEGAL ARG" to "ARG NOT ARRAY"** (36)

Arithmetic
==========

*** Advance Warning: Overflow default will be changed from (OVERFLOW 0) to (OVERFLOW T)** (2617)

In Interlisp-D, the action taken on arithmetic overflow is globally determined by the function OVERFLOW (described below). Currently, the default setting is (OVERFLOW 0), which signifies that arithmetic overflow and division by zero will not cause an error. In a future release, this default will be changed to (OVERFLOW T) so arithmetic overflow and division by zero will cause an error. Users are encouraged to run their programs with (OVERFLOW T), and to change any code which depend on overflow not causing an error.

(OVERFLOW FLG)
Sets a flag that determines the system response to arithmetic overflow and division by zero; returns the previous setting.

For integer arithmetic: If FLG=T, an error occurs on integer overflow or division by zero. If FLG=NIL, the largest integer is returned as the result of the overflowed computation. If FLG=0, the result is returned modulo 2^32 (the default action). If FLG=NIL or 0, integer division by zero returns zero.

For floating point arithmetic: If FLG=T, an error occurs on floating overflow or floating division by zero. If FLG=NIL or 0, the largest floating point number is returned as the result of the overflowed computation or floating division by zero.

*** Advance Warning: (ZEROP X) = (EQ X 0); will be equivalent to (EQP X 0)** (317)

In the Interlisp Reference Manual, (ZEROP X) is defined to be equivalent to (EQ X 0). Some users have complained that this is inconsistent with other lisp dialects, and that (ZEROP 0.0) should not return NIL. In a future release, (ZEROP X) will be equivalent to (EQP X 0). Users who depend on (ZEROP 0.0) returning NIL should change their code to use (EQ X 0).

*** FPLUS, FTIMES call microcode when interpreted** (56)

Previously, the functions FPLUS and FTIMES, when called from the interpreter, didn't go thru the microcoded opcodes but always executed the lisp macrocode.

*** Internal function FTIMES2 no longer defined** (56)

In an old version of the compiler, the function FTIMES was compiled into a call to the function FTIMES2, which has been removed. Some programs compiled in 1982 apparently need recompilation before they will run; if you get UNDEFINED FUNCTION, FTIMES2, you should recompile the offending function.

*** (EXPT 3 -1) returns .333333333 instead of 0** (1581)

The manual states that (EXPT X Y) returns an integer if and only if X is an integer and Y is a non-negative integer.

Processes

*** New process property: BEFOREEXIT used to prevent LOGOUT** (249)

If the process property BEFOREEXIT is the atom DON'T, it will not be interrupted by a LOGOUT. If LOGOUT is attempted before the process finishes, a message will appear saying that Interlisp is waiting for the process to finish. If you want the LOGOUT to proceed without waiting, you must use the process status window (from the background menu) to delete the process.

*** New process property: RESTARTFORM** (566)

If the process property RESTARTFORM is non-NIL, it is the form used if the process is restarted (instead of the original form given to ADD.PROCESS). Of course, the process must also have a non-nil RESTARTABLE prop for this to have any effect.

*** Changes to DISMISS: new arg NOBLOCK** (2208)

(DISMISS MSECSWAIT TIMER NOBLOCK)
If MSECSWAIT and TIMER are both NIL, this is equivalent to (BLOCK). If NOBLOCK is T, DISMISS will not allow other processes to run, but will busy-wait until the amount of time given has elapsed.

*** Control-T does not cause a long DISMISS to return** (2130)

*** WAIT.FOR.TTY spawns mouse if called under the mouse process** (289)

*** ADD.PROCESS property arguments interpreted correctly** (194)

Previously, some combinations of arguments to ADD.PROCESS would be interpreted incorrectly. For example, (ADD.PROCESS <form> 'SUSPEND T) would create a (non-suspended) process with the name SUSPEND.

*** PROCESSPROP can remove last user-defined property from a process** (101)

Previously, only the last property value, not both the name and value, would get removed from the list.

*** RESTART.PROCESS does not hang** (193)

Previously, if RESTART.PROCESS was called on a process which has been created with SUSPEND=T and never started, this would cause Interlisp to hang (hard reset required).

1108 Microcode

*** 1108 microcode available in 4K & 12K versions** (1790 293)

The 1108 hardware is now available with either of two processor boards: the standard board with a 4K microstore, or the Extended Processor Option (CPE) board with a 12K microstore. This does not change the installation or operation of Interlisp --- the Interlisp sysout contains microcode for both the microstore options, and the appropriate one is automatically loaded when Interlisp is started. To provide a visual indication of which size microstore is installed, the 1108 MP display will show 1109 when the 12K microcode is running (instead of 1108).

The 12K microcode contains a number of operations in microcode which were formally implemented in Lisp, so there is a performance improvement. For example, DRAWLINE, BIN, and MAKENUMBER are implemented in the 12K microcode. In the future, any announcements of 1108 microcode changes apply to BOTH microcodes, unless explicitly stated otherwise.

*** 1108 microcode fixes** (1790 1473 1723 2088 2266)

A number of obscure microcode bugs, which could cause intermittent system failures, have been fixed.

*** Pressing 1108 STOP key in RAID will not crash Interlisp** (1482)

In some circumstances, pressing the STOP key when the 1108 was in RAID caused an unrecoverable error, whereas typing control-D would succeed. This was due to a microcode bug.

Library Packages

*** BUSEXTENDER, BUSMASTER: new, prototype packages for using high-speed parallel port on 1108 CPE board** (2290)

The extended 1108 CPE board includes a high-speed parallel port. Currently, hardware for using this parallel port is in development. BUSEXTENDER and BUSMASTER are the prototype versions of the software used for controlling this port. They are being made available to the user community to provide advance information to potential future users.

BUSEXTENDER contains the low-level Interlisp functions used to access the parallel port.

BUSMASTER is an application which uses BUSEXTENDER and special hardware (currently under development) to communicate to IBM PC- or Multibus-compatible peripheral devices.

*** CMLARRAY: the CMLARRAYS file package command now works as advertised** (1039)

*** CMLARRAY: INITIALCONTENTS property works correctly** (1224)

*** COLOR: LOGOUT no longer crashes if color display on** (256)

*** COLOR: (COLORDISPLAY T) no longer breaks with "Illegal arg - NOBIND"** (526)

*** FILEBROWSER: Totally rewritten; many improvements** (836 746 2106)

The most significant changes are:

The Info command has been removed, and the info window has been merged with the browser window. There is a menu of file properties under the main window; this selects the information to be fetched when the Update command is buttoned.

The Rename command now takes a default destination directory when called.

The Copy command now works when you're only copying a single file.

The See command pops up a scrollable window containing the listing of the file. (The old version's window didn't scroll). This window is reused for the next See command if it has been closed.

The file browser has its own prompt window, and no longer pops up superfluous windows.

If you close or shrink a file browser window and there are unexpunged deleted files, an "FB close options" menu will appear, asking whether or not to expunge deleted files before shrinking or closing the window.

The file browser window shrinks to a distinctive "file drawer" icon, which includes the current file browser pattern.

It uses a "nicer" font for the list of files.

Multiple file browsers can "do things" at the same time

Shift-selecting out of a file browser window will shift-select the full name of the file selected. Only one file can be shift-selected at a time.

Can supply new pattern to the filebrowser by middle-buttoning the UPDATE command, and selecting the "New Pattern" option from the menu that pops up.

The file browser window can scroll horizontally, so the user can see all of the properties listed. Above the browser window is a list of colume labels, which scroll horizontally as the browser window does.

*** FTPSERVER: Enumerating files on a remote machine running FTPSERVER now works correctly** (1658)

*** FTPSERVER: COPYFILE to remote 1108 won't cause MP 9318 error** (2163)

COPYFILE to a remote 1108 running FTPSERVER would sometimes cause a serious error. FTPSERVER has been fixed so this will not happen.

*** GRAPHER: Extensively revised; new function HARDCOPYGRAPH; node formatting extended** (1392 2034)

GRAPHER has been extensively revised, so that it uses much less memory space per node. Whereas the old Grapher created a bitmap per node, the new one doesn't. The price is that scrolling may take a little longer. To REDISPLAYW a very large graph takes twice as long as it used to (if you don't like this, set CACHE/NODE/LABEL/BITMAPS/FLG to T). Also, the GRAPHRECORD was changed to use half as many cons cells. This version will not run in Carol or older <lispcore> systems if the user depends on nodefonts being defaulted to the DEFAULTFONT font class.

(HARDCOPYGRAPH GRAPH/WINDOW FILE IMAGETYPE TRANS)
Produces a file from a formated graph (e.g., like SHOWGRAPH, only for files). If GRAPH/WINDOW is a window, HARDCOPYGRAPH will operate on the GRAPH property of the window. If the device field of the file name is LPT, the file will automatically get sent to the appropriate printer. IMAGETYPE is either PRESS or INTERPRESS, and defaults to INTERPRESS. TRANS is a position in screen points of the lower left corner of the graph from the lower left corner of the piece of paper.

(DISPLAYGRAPH GRAPH STREAM CLIP/REG TRANS)
Put the specified graph on STREAM (which can be any image stream) with coordinates translated to TRANS. Some streams might also implement CLIP/REG as a clipping region. This is primarily an efficiency hack for the display.

GRAPHER now allows nodes to be "boxed" with borders of arbitrary shades and widths. Borders work for regular labels and bitmap labels, but not for imageobject labels. The old graphnode field BOXNODEFLG has been renamed NODEBORDER. It takes the following values:

NIL	no border, as before
T	black border, 1 pixel wide, as before
0	no border
1,2,3...	black border of the given width
-1,-2...	white border of the given width
(w s)	where w is a fixp and s is a texture or a shade; yields a border w wide filled with the given shade s.

A new graphnode field, NODELABELSHADE, contains the background shade of the node. This allows GRAPHER to remember when a node is inverted. When a node is displayed, the label area for the node is first painted as specified by NODELABELSHADE, then the label is printed in INVERT mode. This does not apply to labels that are bitmaps or image objects. The legal values for the field are: NIL (same as WHITESHADE), T (same as BLACKSHADE), a texture, or a bitmap.

(RESET/NODE/BORDER <node> <border> <stream>)
(RESET/NODE/LABELSHADE <node> <shade> <stream>)
These functions reset the appropriate fields in the node. If <stream> is a displaystream or a window, the old node will be erased and the new node will be displayed. Both functions take the atom INVERT as a special value for <border> and <shade>. It reads the node’s current border or shade, calculates what would be needed to invert it, and does so.

LAYOUTGRAPH previously used a 1-pixel black box to mark certain nodes in order to indicate where it had snapped links. That is still the default action. However, the appearance of marked nodes can be controlled by adding (MARK ....) to the FORMAT argument of LAYOUTGRAPH. The tail of (MARK ....) is a property list. If the property list is NIL, marking is suppressed altogether. If a BORDER property is specified, the value will be used as the NODEBORDER of marked nodes. If a LABELSHADE property is specified, its value will be used on the marked nodes. Of course, you can specify both a BORDER and NODELABEL property.

LAYOUTGRAPH will read, but not change, the fields NODEBORDER and NODELABELSHADE of the nodes given it (except for the marked nodes, of course). Thus, if one is planning on installing black borders around the nodes after the nodes have been layed out, its a good idea to give LAYOUTGRAPH nodes that have white borders. This will cause the nodes to be layed out far enough apart that when you blacken the borders later, the labels of adjacent nodes will not be overwritten.

When a graphnode is created by the record package, the default values are now taken from the value of the following variables: DEFAULT.GRAPH.NODEBORDER, DEFAULT.GRAPH.NODELABELSHADE, and DEFAULT.GRAPH.NODEFONT. GRAPHER initializes these to NIL. To get the

benefits of this new feature, the user will have to recompile functions that create graphnodes

FLIPNODE now inverts a region that is 1 pixel bigger all around than the node's region. This makes it possible to see black borders after the node has been flipped.

LAYOUTGRAPH takes a new format token. Adding REVERSE/DAUGHTERS to the list of format items will reflect horizontal graphs vertically, and vertical graphs horizontally.

*** LOGOCLOCK process restarts after HARDRESET** (237)

*** SAMEDIR: MIGRATIONS modified: can now have list of directories** (238)

*** SINGLEFILEINDEX: Printing process prevents LOGOUT until finished** (249)

SINGLEFILEINDEX now spawns its process with the process property BEFOREEXIT=DON'T, so that it will not be interrupted by a LOGOUT. If LOGOUT is executed before the process finishes, a message will appear saying that Interlisp is waiting for the process to finish. If you want the LOGOUT to proceed without waiting, you must use the process status window (from the background menu) to delete the process.

*** SINGLEFILEINDEX: new variable \SINGLEFILEINDEX.DONTSPAWN** (294 1850)

If the global variable \SINGLEFILEINDEX.DONTSPAWN = NIL, SINGLEFILEINDEX will spawn a process to process and print the file. If the variable is non-NIL, the processing is done in the current process. When SINGLEFILEINDEX is loaded, \SINGLEFILEINDEX.DONTSPAWN is initialized to NIL if it is not already set.

*** SPY: "recursive merging" reworked, new functions** (2305 1968)

The SPY merge algorithm sometimes produced incorrect results when viewing recursive calls, like functions showing up at 200%. This has been fixed.

The macro WITH.SPY has been added, identical to the inconsistently-named WITH-SPY.

(SPY.LEGEND) creates a window documenting the meaning of the different SPY node types.

(SPY.BUTTON) creates a button which, when touched once turns on SPY, touched again, turns it off and calls (SPY.TREE 10). This is useful for watching what's going on in the system without typing a lot.

*** SYSEDIT: EXPORTS.ALL (loaded by SYSEDIT) does not reset DIRECTORIES** (857)

EXPORTS.ALL contains definitions for system records, and is used to edit system code. Previously, when this file was loaded, it would reset the variables DIRECTORIES and LISPUSERSDIRECTORIES to point to the directories used by the Interlisp-D maintainance group.

*** WHEREIS: Several changes to help users create and maintain their own databases** (126 1625)

Previously, the WHEREIS package interpreted the value of the variable WHEREIS.HASH as the full file name of the single hash file database to search. Now, WHEREIS.HASH is interpreted as a list of hash file names, to be searched in order. This allows the user to keep a number of separate WHEREIS databases for different projects. Also, instead of accepting the hash file filenames as fully-qualified filenames, they are found by searching the directories on DIRECTORIES. WHEREIS.HASH is initialized to NIL.

The function WHEREISNOTICE has also been extended, to help users create and maintain WHEREIS databases:

(WHEREISNOTICE FILEGROUP NEWFLG DATABASEFILE)
Inserts the information about all of the functions on the files in FILEGROUP into the WHEREIS data base contained on DATABASEFILE. If DATABASEFILE is NIL, the first entry on WHEREIS.HASH is used.

FILEGROUP may be simply a list of files, in which case each file thereon is handled directly; but it may also be a pattern to be given as a filegroup argument to DIRECTORY, so &, $, etc. may be used.

If NEWFLG is NIL, the information from the files in FILEGROUP is added to the database DATABASEFILE. If NEWFLG is non-NIL, a new version of DATABASEFILE will be created containing the database for the functions specified in FILEGROUP. If NEWFLG is a number, the hash file will be created with NEWFLG entries. Otherwise, it will be created to allow 20000 entries.

Miscellaneous

*** New variable MAKESYSNAME for identifying Interlisp-D releases**

In the Harmony release sysout, the variable MAKESYSNAME is set to the atom HARMONY. In future releases, this variable will be set to the current release name.

*** For Harmony Release, (LISPVERSION) = 37376**

Previously, the built-in version number was not consistently changed for different releases of Interlisp-D. In future releases, the Interlisp version number will be incremented, and announced in the release message.

*** PROMPTFORWORD revised; doesn't grab TTY; argument renamed** (891 553 1842)

PROMPTFORWORD no longer grabs the tty stream by default. Like READ, if it is called in a process that is not the tty process, it waits for the user to click the mouse in its window, then grabs the tty.

The PROMPTFORWORD argument TIMELIMIT.secs has been renamed URGENCY.OPTION, which is interpreted as follows: If NIL, PROMPTFORWORD quietly wait for input, as READ does; if a number, this is the number of seconds to wait for the user to respond; if T, this means to wait forever, but periodically flash the window to alert the user; if TTY, then PROMPTFORWORD grabs the TTY immediately. When URGENCY.OPTION=TTY, the cursor is temporarily changed to a different shape to indicate the urgent nature of the request.

The last argument to PROMPTFORWORD, OLDSTRING, has been deleted.

Typing control-W now has the normal behavior (delete last word), rather than being a synonym of control-Q (delete all type-in).

PROMPTFORWORD only calls RINGBELLS once to attract the attention of the user.

*** Time-zone variables to control date printout: \TimeZoneComp, \BeginDST, \EndDST** (1077)

These variables are normally set automatically if you have a properly functioning time server on your net. For standalone machines, or old sysouts, you may need to set them by hand (in your init file) if you are not in the Pacific time zone. \TimeZoneComp is the number of hours west of Greenwich (negative if east); \BeginDST is the day of the year on or before which Daylight Savings Time takes effect (i.e., the Sunday on or immediately

preceding this day); \EndDST is the day on or before which Daylight Savings Time ends. Days are numbered with 1 being January 1, and counting the days as for a leap year. In the USA where Daylight Savings Time is observed, \BeginDST = 121 and \EndDST = 305. In a region where Daylight Savings Time is not observed at all, set \BeginDST to 367.

*** (TIMEREXPIRED? X Y) documentation wrong** (1226)

If X and Y are variables whose values are timers, (TIMEREXPIRED X Y) is true if X is set to an EARLIER time than Y. The Reference Manual was wrong: it said that it returned true if X was later than Y.

*** IDATE was wrong in March of leap year -- fixed** (153)

*** GREET now asks for init file in typescript window** (231)

In GREET, if the system can’t find the file {DSK}INIT.LISP, the user is asked to type the name of the site initialization file. Previously, this prompt was printed in the prompt window. Now, the prompt is printed in the top level typescript window.

*** New function NORMALCOMMENTS for setting NORMALCOMMENTSFLG** (2035 2046)

The interface for setting the "remote comment" facility has changed. The recommended way to enable and disable this facility is to call the new function NORMALCOMMENTS, rather than setting the variable NORMALCOMMENTSFLG.

(NORMALCOMMENTS NIL) enables the "remote comment" facility, and (NORMALCOMMENTS T) disables it (the default).