Inferno Release 2.3 Alpha Version
January 1999

What is Inferno?

See the Introduction to the Inferno System at
http://www.lucent-inferno.com/Pages/Developers/Documentation/R2.3/intro23a.PDF.

For more information see the System Overview at
http://www.lucent-inferno.com/Pages/Developers/Documentation/R2.3/SysOver23a.PDF.

For an overview of Security in Inferno see Inferno Security at http://www.lucent-inferno.com/Pages/Developers/Documentation/R2.3/security23a.PDF.

Getting Started

Inferno Release 2.3 Alpha Version is available for four platforms:

- Windows 95 and Windows 98
- Windows NT 4.0
- Solaris/SPARC 2.5.1 or 2.6 and Solaris/x86
- Linux

Note: If previously you have installed an older version of the Inferno system, before you install Inferno Release 2.3 Alpha Version, uninstall any previous version of Inferno so that it does not interfere with this new version.

Download the version for your platform and follow the instructions for your platform.

To install on Windows 95/98 or NT platforms:

Use your browser to download the InstallShield executable selection, inferno.exe. Use the Save As function to direct the file to an empty folder.

Run inferno.exe. It will extract itself into several files. Run the file called setup.exe to install the Inferno system. The installation procedure is automatic except for a couple of dialog boxes.

For this Inferno Release 2.3 Alpha Version, `setup.exe` will make an entry for **Inferno 2.3 Emulator Alpha** on your **Start >> Programs** menu. Except for such shortcuts, the distribution will go into the folder `C:Program Files\Inferno\Inferno2.3Emulator` by default.

To run Inferno, select the **Emulator** subentry from the **Start >> Programs >> Inferno 2.3 Emulator Alpha** menu. The command-line or text window for the Inferno control console displays.

Note: The path to the Inferno emulator on Windows is `C:Program Files\Inferno\Inferno2.3Emulator\Nt\386\bin\emu.exe`. However, for this Alpha Release you must use the entry on the **Start** menu to start Inferno Release 2.3 Alpha Version.

When you start the emulator, a window opens with a few lines of text and presents you with the system prompt, *<machinename>$*:

```
Inferno Release 2.3 Build 75i (<date> <time>) main (pid=<pid>) interp
Initialize Dis: /dis/sh.dis
<machinename>$
```

This **EMU** window is the Inferno control console.

At the **EMU** control console prompt (*<machinename>$*), type the following lines. Press **Enter** or **Return** after each line:

```
<machinename>$ bind '#I' /net
<machinename>$ lib/cs
<machinename>$ wm/logon
```

You will see the Inferno Window Manager Logon Screen. Enter `inferno` as the user Name and press **Enter**. The first time you log on as user `inferno`, you will see the Inferno License Agreement Acceptance window. Read the agreement and click on the **I Accept** button to begin using your Window Manager session.

To run the Charon Browser, open the Inferno menu by selecting the Inferno button at the lower left corner of the Inferno window. Select **Applications** and then select the **Charon Browser** item. For further information, please see the Charon Web Browser section below.

To install on UNIX-like systems:

Download the appropriate files, Inferno_Base.tar.gz and Inferno_X.tar.gz, where X is one of Solaris_sparc, Solaris_386, or Linux_386.

Put the files in an empty directory such as /usr/inferno and extract it. If you use tar, remember to set the p flag to preserve permissions.

To run Inferno, use the appropriate binary for the system you are on:

```
/usr/inferno/Solaris/sparc/bin/emu
/usr/inferno/Solaris/386/bin/emu
/usr/inferno/Linux/sparc/bin/emu
```

When you run this command line in a window, that window becomes the control console for the Inferno emulator.

When you start the emulator, a window opens with a few lines of text and presents you with the system prompt, *<machinename>$*:

```
Inferno Release 2.3 Build 75i (<date> <time>) main (pid=<pid>) interp
Initialize Dis: /dis/sh.dis
<machinename>$
```

This **EMU** window is the Inferno control console.

At the **EMU** control console prompt (*<machinename>$*), type the following lines. Press **Enter** or **Return** after each line:

```
<machinename>$ bind '#I' /net
<machinename>$ lib/cs
<machinename>$ wm/logon
```

You will see the Inferno Window Manager Logon Screen. Enter inferno as the user Name and press **Enter**. The first time you log on as user inferno, you will see the Inferno License Agreement Acceptance window. Read the agreement and click on the **I Accept** button to begin using your Window Manager session.

To run the Charon Browser, open the Inferno menu by selecting the Inferno button at the lower left corner of the Inferno window. Select **Applications** and then select the **Charon Browser** item. For further information, please see the Charon Web Browser section below.

Using the Inferno Emulator, EMU:

See Using the Inferno System at
http://www.lucent-inferno.com/Pages/Developers/Documentation/R2.3/getstart23a.PDF.

What's New?

Inferno Release 2.3 Alpha Version is significantly different from the Inferno Release 2.0 product, which was the last public release of the Inferno operating system.

Changes include efficiency and performance improvements, and many bug fixes including the elimination of memory leaks. Much of the change has been to the source code for the product. While these changes are visible only to those who have a source code license, the effects of those changes provide a faster, more reliable, and a more stable system.

Significant functionality changes include:

- a completely new web browser, **Charon**, with a new look, much faster performance, and compliance with the HTML 3.2 standard [SEE *Charon Web Browser* section below]

- a new shell, known as "mash" and which functions both as a programmable shell and as a build tool with functionality similar to make. You can access **Mash** using the **Development Tools** menu. [*SEE Mash Shell Application* section below]

- an updated Limbo compiler with improved type checking and warnings

- easier administration of Inferno servers, including more advanced logging facilities

- easier creation and maintenance of synthetic file systems because of a new set of libraries

- updated interfaces to the security module and new modules that ease the authentication process

- new self referencing modules with the ability to pass this reference to other modules (This functionality is used extensively in mash, Charon, and the new styx libraries.) [SEE *Self Module Pointer* section below]

- new Sys module routine, Sys->aprint, which is similar to Sys->sprint, but which returns an array of bytes instead of a string

- new Draw module routine, Draw->Display.cursor() that sets the current cursor associated with the Display

- new Tk module function, TK->windows() that returns a list of the TK->Toplevel adts, one for each window on the screen

- new Wmlib module function, Wmlib->untaskbar() that will restore a window from the task bar

- new Bufio module routine, Bufio->Iobuf.setfill() that associates a BufioFill module with an Iobuf (BufioFill defines a single routine named fill() that Bufio calls when it needs more data for a read request.)

- new **xd** hex dump command that dumps standard input or the contents of a file to standard output as a series of hexadeximal digits with each line of output representing 16 bytes (Preceding the bytes, the address of the first byte in the line is printed and after the bytes, a 16-character field is printed with "." representing an unprintable character. Type **xd -?** for the usage message.)

- more reliable #s device used with Sys->file2chan

- updated port(s) to Linux/x86 and Solaris/x86

- updated Dis interpreter with improved garbage collection that is more efficient and reliable

- new process monitoring and logging features for Inferno servers [*SEE New Server Process Monitoring and Logging Features* section below]

- the **Media Players** menu item for the **GIF Viewer** is now labeled the **Image Viewer** since you can view gif, jpg, and jpeg files

- new **Stopwatch** application that you can access from the Inferno **Applications** menu

- new **Telnet** application that you can access from the Inferno **Applications** menu

- new Chat Server application. See the information at http://www.lucent-inferno.com/Pages/Developers/Demos/chatapp.html

- new **Demos** menu item on the **Inferno** menu gives you access to the **Coffee**, and new **Linpack**, and **Colors** demos (When you select a color in the Colors demo, the associated RGB values are shown. The Linpack Demo is an example of linear algebra routines working in Limbo.)

- new **Window Controls** utility on the **System** menu provides the ability to raise or lower the focus of a window, or move, delete, or hide (minimize) a window.

- new **Wish** menu item on the **Development Tools** menu allows you to create Tk items

- the **Plumb** or **Plumbing** window that was on the task bar for Inferno R 2.0 has been replaced by a more general message window that is labeled **Console**

Charon Web Browser

The Charon Web Browser is a completely new web browser. It uses the Inferno Limbo Draw and is faster, and it implements all the standard HTML 3.2 features.

Charon has a new look, see http://www.lucent-inferno.com/Pages/Developers/Documentation/R2.3/Charon.PDF.

Mash Shell Application

The new **mash** shell functions as a programmable shell and as a build tool that is similar to **make**. When you are in EMU,

you can access Mash from the Inferno **Development Tools** submenu.

The Inferno Mash Manual is at
http://www.lucent-inferno.com/Pages/Developers/Documentation/Mash/mashman.html.

You can see the Inferno Tk "builtins" Manual at
http:// www.lucent-inferno.com/Pages/Developers/Documentation/Mash/tkman.html.

A Mash Tutorial is at
http://www.lucent-inferno.com/Pages/Developers/Documentation/Mash/mashtut.html.

A Make Tutorial is at
http://www.lucent-inferno.com/Pages/Developers/Documentation/Mash/maketut.html.

Self Module Pointer

New to R2.3 is the ability for a module to load a pointer to itself. The syntax for this is the same as that for loading a builtin module, but using the special builtin module name "$self". The module include file `sys.m` contains the definition:

```
SELF: con "$self"; #Language support for loading my instance
```

so you can write something like:

```
mymod := load Mymod SELF;
```

Then you can pass the module pointer `mymod` to another module that is expecting a module of type `Mymod`. Note that this is quite different from writing:

```
mymod := load Mymod Mymod->PATH;
```

since `$self` gives you a pointer to the actual module instance that you are executing, whereas loading the module again gives you a fresh instance with a different set of module data. Using `$self`, the module routines that we pass `mymod` to can access the interface defined by `Mymod`, and use it to access our state.

A particularly useful aspect of `$self` is that the type specified in the load statement does not have to be the full type that the executing module was defined with. For example, module `bufio.m` defines two modules called `BufioFill` and `ChanFill`. Both export a function called "`fill`" (with the

same type), but ChanFill adds an "init" routine. In ChanFill->init(), we find the code:

```
if (myfill == nil)
     myfill = load BufioFill SELF;
```

This gives ChanFill a restricted version of its interface, which it can pass to the routine Bufio->Iobuf.setfill() that expects a module pointer of type BufioFill (not ChanFill).

Without $self, we would be forced to put the type of the init routine into BufioFill, forcing all instances of this module to be initialized from the same set of arguments. $self allows us the flexibility to initialize different BufioFill instances in different ways, letting the specific application decide what parameters it needs. Similarly, we could add other routines to a common module interface; initialization is just one example of a very powerful and general mechanism.

Display Current Cursor Draw Routine

The Draw module adds the routine Draw->Display.cursor() that allows the setting of the current cursor associated with the Display. Sample usage:

```
d := ctxt.display;
d.cursor(img, hotpt);
```

where img is a 16x32, ldepth 0 Image that specifies the foreground and mask, and hotpt is a Point that specifies the offset of the Image from the current cursor position.

New Server Process Monitoring and Logging Features

lib/srv:

There is an enhanced version of lib/srv that incorporates several new features.

- A task monitor that will watch the running servers and restart them if they fail.
- Ability to write to a log file using either a simple file or the lib/logsrv server.

- New keyword arguments that can be used to control the default functions of the servers startup and the log server.

lib/logsrv:

There is log server process that provides new features for gathering information on the running servers and can be used by external processes for reporting.

The log server contains three distinct file-reporting mechanisms:

1. A log file.
2. A time stamped journal file.
3. A measurements file.

The journaling and measurements functions are supported by user-defined processes that are bound in the namespace to the `/dis/lib` directory containing the `logJournal.dis` and `logMeasure.dis` processes respectively.

Other Documentation

The Inferno Reference Manual is at
http://www.lucent-inferno.com/Pages/Developers/Documentation/Ref_man20/index.html.

The Inferno Limbo Reference Manual is at
http://www.lucent-inferno.com/Pages/Developers/Documentation/Limbo.Rel20/index.html.