Installing the Inferno Software

Vita Nuova
support@vitanuova.com
12 June 2003

Inferno can run as either a native operating system, in the usual way, or as a hosted virtual operating system, running as an application on another operating system. This paper explains how to install Inferno from the distribution media to a hosted environment and how to configure the system for basic networking.

Inferno can run as a hosted virtual operating system on top of Plan 9, Unix or Windows. In this paper, the term Unix is used to cover all supported variants, currently FreeBSD, Linux, HP/UX, Irix and Solaris, and the term Windows covers Microsoft Windows (98, Me, Nt, 2000, and XP). (Windows 98 might first require installation of the Unicode layer update from Microsoft.)

1. Preparation

You should ensure at least 150 Mbytes of free space on the filesystem. The installation program will copy files from the distribution CD to a directory on the filesystem called the inferno_root directory. You can choose the location of this directory. If you are installing to a multiuser filesystem outside your control a subdirectory of your home directory might be most sensible. If you plan to share the Inferno system with other users then common choices for inferno_root are /usr/inferno on Unix and Plan 9 systems, and c:\inferno on Windows systems. Where these appear in examples in this paper you should substitute your own inferno_root directory.

" Step 1a: Choose the inferno_root directory.

Ensure that the user who will run the installation program has appropriate filesystem permissions to create the inferno_root directory and files and subdirectories beneath it.

2. Copying Files

On all platforms the files will be owned by the user doing the installation, except for installation onto a FAT file system (eg, on Windows), where the files appear to be owned by Everyone because FAT does not record ownership.

" Step 2a: Insert the distribution CD into the CD drive.

On Unix and Plan 9, mount the CD to a suitable location on the filesystem, call this location cd_path. On Windows, note the drive letter of the CD, call this drive letter cd_drive. The files will be copied by an Inferno hosted installation program which runs directly from the CD. The directory /install on the CD contains an installation program for each supported platform a shell script for Unix and Plan 9 and an executable for Windows. The Plan 9 install script is called Plan9.rc and determines the CPU type from the environment variable cputype . The Unix install scripts all have names of the form host_os-host_arch.sh where host_os will be one of: FreeBSD, Linux , or Solaris and host_arch will be one of: 386, mips , power or sparc . Most platforms offer just the one obvious combination. The Windows installation program is called setup.exe ; it is used on all varieties of Windows. The next step describes how to begin the installation by running the program that corresponds to your host system.

" Step 2b: Run the installation script.

The installation program will copy files from the CD to the filesystem. The Windows installation program will also create registry entries and add an Inferno item to the Windows start menu. On Plan 9, run

```
rc cd_path/install/Plan9.rc inferno_root
```

```
; lc /
FreeBSD/        Unixware/       icons/          libkern/        man/            prof/
Hp/             acme/           include/        libkeyring/     mkconfig        prog/
Inferno/        appl/           keydb/          libmath/        mkfile          services/
Irix/           asm/            legal/          libmemdraw/     mkfiles/        tmp/
LICENCE         chan/           lib/            libmemlayer/    mnt/            tools/
Linux/          dev/            lib9/           libtk/          module/         usr/
MacOSX/         dis/            libbio/         licencedb/      n/              utils/
NOTICE          doc/            libcrypt/       limbo/          net/            wrap/
Nt/             emu/            libdraw/        locale/         nvfs/
Plan9/          env/            libfreetype/    mail/           o/
Solaris/        fonts/          libinterp/      makemk.sh       os/
;
```

Only the files and directories in and below the inferno_root directory on the host filesystem are immediately visible to an Inferno process; these files are made visible in the root of the Inferno file namespace. If you wish to import or export files from and to the host filesystem you will need to use tools on your host to move them in or out of the Inferno visible portion of your host filesystem (see the manual pages os(1) and cmd(3) for an interface to host commands). (We plan to make such access direct, but the details are still being worked out.) From this point onwards in this paper all file paths not qualified with inferno_root are assumed to be in the Inferno namespace. Files created in the host filesystem will be created with the user id of the user that started emu and on Unix systems with that user's group id.

4. Setting the site's time zone

Time zone settings are defined by files in the directory /locale. The setting affects only how the time is displayed; the internal representation does not vary. For instance, the file /locale/GMT defines Greenwich Mean Time, /locale/GB-Eire defines time zones for Great Britain and the Irish Republic (GMT and British Summer Time), and /locale/US_Eastern defines United States Eastern Standard Time and Eastern Daylight Time. The time zone settings used by applications are read (by daytime(2)) from the file /locale/timezone, which is initially a copy of /locale/GB-Eire. If displaying time as the time in London is adequate, you need change nothing. To set a different time zone for the whole site, copy the appropriate time zone file into /locale/timezone:

```
cp /locale/US_Eastern /locale/timezone
```

To set a different time zone for a user or window, bind(1) the file containing the time zone setting over /locale/timezone, either in the user's profile or in a name space description file:

```
bind /locale/US_Eastern /locale/timezone
```

5. Running the Window Manager wm

Graphical Inferno programs normally run under the window manager wm(1). Inferno has a simple editor, wm/edit, that can be used to edit the inferno configuration files. The `power environment' for editing and program development is acme(1), but rather that throwing you in at the deep end, we shall stick to the simpler one for now. If you already know Acme from Plan 9, however, or perhaps Wily from Unix, feel free to use Inferno's acme instead of edit.

```
" Step 5a: Start the window manager.
```

Invoke wm by typing

```
wm/wm
```

You should see a new window open with a blue-grey background and a small Vita Nuova logo in the bottom left hand corner. Click on the logo with mouse button 1 to reveal a small menu. Selecting the Edit entry will start wm/edit. In common with most wm programs the editor has three small buttons in a line at its top right hand corner. Clicking on the X button, the rightmost button, will close the program down. The leftmost of the three buttons will allow the window to be resized after clicking it drag the window from a point near to either one of its edges or one of its corners. The middle button will minimise the window, creating an entry for it in the application bar along the bottom of the main wm window. You can restore a minimised window by clicking on its entry in the application bar. The initial wm configuration is determined by the contents of the shell script /lib/wmsetup (see toolbar(1) and sh(1)).

Now the server is ready but we need a client.

Either use a third machine or (more likely at first) simply start another emu instance in a new window. Start its connection server, again by typing

```
ndb/cs
```

The connection server is fundamental to the Inferno network. Once networking is set up, when subsequently starting up a client you should start cs before starting the window system. Note that if you are running the Inferno instance as a server, or combined server and client, the svc/net that starts the network services automatically starts cs, and you need not do so explicitly.

" Step 12b: Generate a client certificate.

Obtain a certificate for the client in the same way as on the server. We shall obtain a certificate for use with a specific server by storing it in a file whose name exactly matches the network address of the server

```
getauthinfo tcp!hostname
```

Use the current machine's hostname. Getauthinfo stores the certificate in the file /usr/user/keyring/keyname where user is the name in /dev/user and keyname is the argument given to getauthinfo. Again, answer yes to the question that asks if you want to save the certificate in a file.

Now that both client and server have a certificate obtained from the same signer it is possible to establish a secure connection between them. Note that getting keys and certificates with getauthinfo is normally done just once (or at most once per server when the default key is not used). In short, all the work done up to now need not be repeated. After this, provided the keys were saved to a keyring file, as many authenticated connections can be made as desired until the certificate expires (which by default is whenever the password entry was set to expire). Also note that the certificates for different machines can have different signers, and one can even use different certificates for the same machine when the remote user name is to differ (the -f option of getauthinfo can then be useful to force an appropriate keyring name).

" Step 12c: Make an authenticated connection.

The script svc/net on the server started fundamental name services and also a Styx file service. That can also be started separately using svc/styx. In either case the namespace that is served is the one in which the command was invoked. Now you can test the service.

Confirm that /n/remote is an empty directory by typing

```
lc /n/remote
```

You can now mount the server's name space onto the client's directory /n/remote by typing

```
mount serveraddr /n/remote
```

Where serveraddr is the IP address of the server or a name that the host can resolve to the IP address of the server. Now

```
lc /n/remote
```

should reveal the files and directories in the namespace being served by the server. Those files are now also visible in the namespace of your shell. You will notice that these changes only affect the shell in which you ran the mount command other windows are unaffected. You can create, remove or modify files and directories in and under /n/remote much as you can any other file or directory in your namespace. In fact, in general, a process does not need to know whether a file actually resides locally or remotely. You can unmount the mounted directory with unmount. Type

```
unmount /n/remote
```

You can confirm that it has gone by running

```
ls /n/remote
```

All connections made by Inferno are authenticated. The default connection made by mount is authenticated but uses neither encryption nor secure digests. You can pass an argument to mount to specify a more secure connection: its -C option gives it a hashing and an encryption algorithm to be applied to the connection.