# Installation of the Inferno Software

Inferno can run as either a native operating system or as a *hosted* virtual 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.

Third edition 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 and Solaris, and the term *Windows* covers Microsoft Windows 95, Windows 98, Windows Nt and Windows 2000.

## 1. Preparation

You should ensure that there are 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 a subdirectory of your home directory might be a sensible choice of location. 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:\users\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 except Windows 95 and Windows 98, the files copied will be owned by the user who runs the installation. On Unix, Windows Nt and Windows 2000, if the installation is run by user root and a user inferno exists the installed files will be owned by user inferno .

¡ **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 . Not all combinations are available. The Windows installation program is called Nt-386.exe and can be used on Windows Nt, Windows 95, Windows 98 and Windows 2000 - it is only available in combination with the 386 CPU architecture option. 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
```

Where *inferno_root* is the path to the chosen Inferno root directory. The CPU architecture will be inferred from the environment variable cputype. On Unix, run

```
sh cd_path/install/host-os-host_arch.sh inferno_root
```

Where *host_os* is the Unix variant name (FreeBSD, Linux or Solaris). *host_arch* is the CPU type (most likely 386), and *inferno_root* is the path to the chosen Inferno directory. On Windows, run

```
cd_drive:\install\setup.exe
```

The Windows installation program will ask you to choose the location of the installation directory on the hard disk.

Files will then be copied from the CD to the *inferno_root* subtree on the filesystem. Inferno is now installed, but some configuration is required.

## 3. Running Inferno

Inferno host executables are all kept in a single directory corresponding to the combination of host operating system and CPU architecture — the Inferno bin directory.

```
inferno_root/host_os/host_arch/bin
```

Plan 9 users should add a line to their lib/profile file that binds this directory after their /bin directory.

```
bind -a /usr/inferno/Plan9/$cputype/bin /bin
```

The bind is done after the existing *bin* directory to avoid hiding your existing Plan 9 compilers. If, at a later stage, you build either the hosted or native Inferno kernels you should ensure that the Inferno compilers are used rather than the Plan 9 compilers. The differences between the two compiler suites are likely to be reduced in future releases. On Windows systems the *host_os* is always Nt and *host_arch* is always 386 and the installation program will create an entry on the *start menu* to invoke Inferno. For Unix systems or Windows systems in which Inferno will be started from a command shell, the environment variable PATH should be set to include the Inferno bin directory. For Windows 95 and Windows 98 this should be done in the \autoexec.bat file by adding a line like

```
PATH="c:\users\inferno\Nt\386\bin";%PATH%
```

You will need to reboot Windows to have the system re-read the \autoexec.bat file. For Windows NT and Windows 2000 modify the Path environment variable through *Control Panel -> System -> Environment*. For Unix this should be done in your .profile file by adding a line like

```
PATH="/usr/inferno/Linux/386/bin:$PATH"
```

Don't forget to ensure that PATH is exported. You may need to log out and back in again for these changes to take effect.

¡ **Step 3a: Start Inferno.**

Hosted inferno is run by invoking an executable called *emu*.

On Windows, select the Inferno option from the *start menu*. This will invoke *emu* with appropriate arguments to find its files in *inferno_root*. If you need to change any of the options passed to *emu* when invoked from the *start menu* you need to do this by clicking the right mouse button on the Windows taskbar and choosing *Properties -> Start Menu Programs -> Advanced* to modify the shortcut used for Inferno. For Unix and Plan 9, you will need to tell *emu* where to find the Inferno file tree by passing it the -r *rootpath* command line option. For example

```
emu -r/usr/john/inferno
```

Without the -r option it will look for the file tree in /usr/inferno on Plan 9 and Unix and \users\inferno on the current drive for Windows. By default, when using graphics *emu* will use a window with a resolution of 640 x 480 pixels. To use a larger resolution you will need to pass *emu* the -g *Xsize*x*Ysize* command line option. So, for example, to invoke *emu* as above but with a resolution of 1024 x 768 pixels the full command line would be

```
emu -r/usr/john/inferno -g1024x768
```

When invoked in this way *emu* displays a command window running the Inferno shell /dis/sh.dis. To avoid typing the command line options each time you invoke *emu* you can store them in the environment variable EMU which is interrogated when *emu* is started and might as well be set along side the PATH environment variable if the same configuration options are to be used on each invocation.

```
set EMU="-rd:\users\john\inferno -g1024x768"
```

for Windows.

```
EMU=(-r/usr/john/inferno -g1024x768)
```

for Plan 9, and

```
EMU="-r/usr/john/inferno -g1024x768"
```

for Unix. An alternative to using the EMU environment variable is to place the correct invocation in a script file (or batch file, for Windows) and invoke that instead of running *emu* directly. It is important to note that for Windows the -r option also serves to indicate both the drive and directory on to which the software has been installed. Without a drive letter the system will assume the current drive and will fail if the user changes to an alternative drive. Once the environment variables or scripts are set up, as described above, invoking

```
emu
```

or the appropriate script file, should result in output which looks something like this

```
Inferno Third Edition (4 May 2000) main (pid=112260) interp
Initialize Dis: /dis/emuinit.dis
;
```

You are now running third edition Inferno, built on May 4th 2000 and the word interp means that the Inferno Dis virtual machine is interpreting Dis instructions. You can add a further option -c1 (see the *emu* (1) manual page) to cause *emu* to start up in *compile* mode in which the JIT compiler is automatically invoked to compile all Dis operations to native machine instructions on demand. In *compile* mode most programs will run significantly faster. Whether in compile or interpreted mode you should now have a functional hosted Inferno system. When Inferno starts the initial /dis/sh.dis it reads commands from the file /lib/sh/profile before becoming interactive. See the manual pages for the shell *sh* (1) to learn more about tailoring the initial environment. The semicolon is the default shell prompt. From this command window you should be able to see the installed Inferno files and directories

```
lc /
```

The command *lc* presents the contents of its directory argument in columnar fashion to standard output in the command window.

```
; lc /
Inferno/    emu/       lib/        mkfile     services/
Linux/      fonts/     lib9/       mkfiles/   tk/
Nt/         icons/     libbio/     module/    tmp/
appl/       image/     libregexp/  n/         tools/
asm/        include/   limbo/      net/       usr/
chan/       interp/    math/       nvfs/      utils/
crypt/      kern/      memimage/   os/
dev/        keyring/   memlayer/   prefab/
dis/        kfs/       mkconfig    prog/
;
```

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). 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 users group id.

## 4. Running the *wm* Window Manager

Many Inferno programs run under the *wm* window manager. Inferno has a simple editor, *wm/edit* , that can be used to edit the inferno configuration files.

¡ **Step 4a: Start the window manager.**

Invoke *wm* by typing

```
wm/wm
```

You should see a new window open with a blue 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 right most button, will close the program down. The left most 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 *sh* (1)).

¡ **Step 4b: Open a shell window.**

Choose the *shell* option from the menu to open up a shell window. The configuration of Inferno will be done from this shell window.

### 5. Manual Pages

Manual pages for all of the system commands are available from a shell window. Use the *man* or *wm/man* commands. For example,

```
man wm
```

will give information about *wm*. And

```
man man
```

will give information about using *man*. *Wm/man* makes use of the Tk text widget to produce slightly more attractive output than *man*. Here, and in other Inferno documentation you will see references to manual page entries of the form *command* (*section*). You can display the manual page for the command by running

man *command*

or

man *section* *command*

if the manual page appears in more than one section.

### 6. Initial Namespace

The initial Inferno namespace is built by placing the root device #/ (see *root* (3)) at the root of the namespace and binding

i) the host filesystem device #U (see *fs* (3)) containing the *inferno_root* subtree of the host filesystem at the root of the Inferno filesystem,

ii) the console device #c (see *cons* (3)) in /dev ,

iii) the prog device #p (see *prog* (3)) onto /prog ,

iv) the IP device #I (see *ip* (3)) in /net , and

v) the environment device #e (see *env* (3)) at /dev/env .

You can see the sequence of commands required to construct the current namespace by running

```
ns
```

To use IP networking, the IP device *ip* (3) must have been bound into /net . Typing

```
ls -l /net
```

(see *ls* (1)) should result in something like

```
dr-xr-xr-x   I   0 bootes  bootes  0 Feb 02 14:48  /net/tcp
dr-xr-xr-x   I   1 bootes  bootes  0 Feb 02 14:48  /net/udp
```

We can now configure Inferno to allow secure connections to other Inferno systems. First we will start a signer process which will generate certificates for both the system making a connection and the system receiving the connection. If there is only one machine in your Inferno network then the same machine will be user as client, server and certificate signer.

### 7. Connection Server

The connection server (see *cs* (8) and *db* (6)) translates symbolic network names and services into instructions for connecting to a given service. The file /services/cs/db defines a mapping from machine names of the form $*service* to either a network address or a numeric address. The connection server will convert such a machine name using this file into either a numeric address or a machine name. A typical /services/cs/db file will look something like this

```
$SIGNER        doppio
$FILESERVER    rotta
$MAILSERVER    doppio
$PROXY         pox
```

Network and service names are passed through to the host for conversion to numeric addresses and port numbers. If the host is unable to convert a service name the connection server will attempt to convert the name using mappings of service and protocol names to Internet port numbers in the file /services/cs/services which should contain at least the following entries

```
styx         6666/tcp             # Main file service
mpeg         6667/tcp             # Mpeg stream
rstyx        6668/tcp             # Remote invocation
infdb        6669/tcp             # Database connection
infweb       6670/tcp             # inferno web server
infsigner    6671/tcp             # inferno signing services
infcsigner   6672/tcp             # inferno signing services
inflogin     6673/tcp             # inferno login service
virgil       2202/udp   virgild   # inferno info
```

For the moment, leave this file as it is. You will need to modify this file, at some point in the future, if you add new services to Inferno and want to refer to them by name.

¡ **Step 7a: Start the connection server.**

To run the connection server, type

```
lib/cs
```

You should now see a new file in the /net directory called cs. Run the command

```
ls /net
```

You should see the following output

```
/net/cs
/net/tcp
/net/udp
```

Before an Inferno machine can establish a connection to an Inferno service on another machine, each needs to obtain a certificate from a common signer. To bootstrap this process we will configure this machine as a signer.

## 8. Network Services

The command *srv* is used to start listeners for local network servers (see *srv* (8)). *Srv* starts a listener for each service configured in /services/server/config.

```
S infsigner    tcp  /dis/lib/signer.dis
S infcsigner   tcp  /dis/lib/countersigner.dis
S inflogin     tcp  /dis/lib/logind.dis
S styx         tcp  /dis/lib/styxd.dis    nossl clear sha md5 rc4 sha/rc4
S rstyx        tcp  /dis/lib/rstyxd.dis   nossl clear sha md5 rc4 sha/rc4
S infdb        tcp  /dis/lib/dbsrv.dis    nossl clear sha md5 rc4 sha/rc4
S virgil       udp  /dis/lib/virgild.dis
M 67           udp  /dis/lib/bootp.dis
M 69           udp  /dis/lib/tftpd.dis
```

Documentation for these servers can be found in section 8 of the Programmers Manual (Volume 1). Start the listeners on this machine by running *srv*.

¡ **Step 8a: Start the network listener services.**

Type

```
lib/srv
```

Your servers will now be running. To confirm this type

```
netstat
```

*Netstat* prints information about network connections. You should see several lines of output, each one describing an announced TCP or UDP service. Depending upon the contents of your /services/server/config file you should see something which looks like this

```
tcp/0    inferno    200.1.1.89!6669         0.0.0.0!0        Announced
tcp/1    inferno    200.1.1.89!6668         0.0.0.0!0        Announced
tcp/2    inferno    200.1.1.89!6666         0.0.0.0!0        Announced
tcp/3    inferno    200.1.1.89!6673         0.0.0.0!0        Announced
tcp/4    inferno    200.1.1.89!6672         0.0.0.0!0        Announced
tcp/5    inferno    200.1.1.89!6671         0.0.0.0!0        Announced
udp/0    inferno    200.1.1.89!2202         0.0.0.0!0        Announced
```

Each line corresponds to a network connection: the connection name, the name of the user running the server, the address of the local end of the connection, the address of the remote end of the connection, and the connection status. The connection name is actually the protocol and conversation directory in /net. The connection addresses are all of the name *host*!*port* for these IP based services, and the remote addresses are not filled in because they all represent listening services that are in the Announced state. In this example the fourth line shows a TCP service listening on port 6673. Examining /services/cs/services with grep (see *grep* (1)) shows that the listener on port 6673 is the Inferno login service.

```
grep 6673 /services/cs/services
```

gives

```
inflogin      6673/tcp            # inferno login service
```

Before the signing service can be used to generate certificates some configuration is required. The signer needs a public and private key-pair and a database of user names and passwords.

## 9. Configuring a Signer

To use authenticated connections we need to setup a signer to generate certificates for users (see *createsignerkey* (8) and *signer* (8)). For two machines to communicate securely they must both have obtained a certificate from the same signer. Choose an Inferno machine to become the signer. If this is the first or only Inferno machine on your network then make this machine the signer.

¡ **Step 9a: Generate a signer key.**

On the signer machine run

lib/createsignerkey *name*

In place of *name* enter the network name of the signer. This value will appear as the signer name in each certificate generated by the signer. *Createsignerkey* creates public and private keys that are used by the signer when generating certificates.

¡ **Step 9b: Enter user names and secrets.**

For each user to be authenticated by the signer run

```
changelogin username
```

You will be prompted to supply a secret (i.e. password) and expiration date. The expiration date will be used as the expiration date of certificates generated for that user. *Changelogin* (see *changelogin* (8)) will create and update entries in the file /keydb/password. For the signer to generate a certificate there must be at least one entry in the password file. If you are not sure at this stage of the names of the users that you want to authenticate then create an entry for the user inferno.

## 10. Establishing a Secure Connection

To establish a secure connection between two machines, each needs to have a certificate signed by a common signer (which need not be a third machine). If you have only one Inferno machine you can experiment with secure connections by making the same machine signer, server and client.

¡ **Step 10a: Generate a server certificate.**

On the server machine, ensure that lib/cs is running. You will need it if you refer to your signer by hostname instead of IP address. If in doubt, or if it is not running, type

```
lib/cs
```

On the server machine, use *getauthinfo* (8) to obtain a certificate and save it in a file named default by running

```
getauthinfo default
```

*Getauthinfo* will prompt for the address of your signer and for a remote username and password combination. *Getauthinfo* will connect to the *inflogin* service on the signer and authenticate you against its user and password database, /keydb/password, using the username and password that you entered. Answer Yes to the question that asks if you want to save the certificate in a file. *Getauthinfo* will save a certificate in the file /usr/*user*/keyring/default where *user* is the name in /dev/user.

¡ **Step 10b: Generate a client certificate.**

Obtain a certificate for the client in the same way. Type

```
getauthinfo default
```

If you wish you can 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
```

*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. If you have only one Inferno system you can still test the configuration by using the same machine as both client and server. Alternatively, start a second copy of *emu* on the same machine and treat one as the server and one as the client.

¡ **Step 10c: Make an authenticated connection.**

On the server, make sure that *srv* is running, type

```
netstat
```

you should see a line for the TCP service listening on port 6666.

```
tcp/2      inferno      200.1.1.89!6666          0.0.0.0!0         Announced
```

If you do not see any output, start *srv* by running

```
lib/srv
```

The listener on port 6666 is expecting to serve the Inferno file protocol *Styx* to export a namespace.

```
grep 6666 /services/cs/services
```

Gives

```
styx          6666/tcp             # Main file service
```

The line for the *styx* service in the file /services/server/config shows that the server/dis/lib/styxd.dis is listening on port 6666. The namespace that *styxd* serves is the one that it inherited when it was started with *srv* . On the client, make sure that *Cs* is running by typing

```
lib/cs
```

Now that lib/cs is running on the client and lib/srv is running on the server, you can test the service. On the client, confirm that /n/remote is an empty directory with

```
lc /n/remote
```

On the client, you can now mount the namespace that *styxd* is serving on the server onto the client directory /n/remote by typing

```
mount tcp!serveraddr /n/remote
```

Where *serveraddr* is the IP address of the server or a name which 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 *styxd* . 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 processes 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 does not use SSL. The arguments passed to *styxd* in its entry in /services/server/config

```
S rstyx        tcp  /dis/lib/rstyxd.dis     nossl  clear  sha  md5  rc4  sha/rc4
```

define the different combinations of security algorithms that *styxd* is prepared to accept. You can pass an argument to *mount* to specify a more secure connection. The -C option to *mount* can be used to specify a hashing and an encryption algorithm to be applied to the connection.

¡ **Step 10d: Make a secure authenticated connection.**

For example,

```
mount -C sha/rc4 tcp!serveraddr /n/remote
```

will make an authenticated connection to the machine given by *serveraddr* using SHA hashing for message digesting and RC4 for encryption and mount the namespace served by the server's *styx* service on the client directory /n/remote.

### 11. Adding new users

Every inferno process has an associated *user name*. At boot time the user name is set equal to your login name on the host operating system. You can change the user name by writing the new user name to the file /dev/user. This is a one time operation, once the user name has been changed from the original it cannot be changed again in that process. The user name is used to select the directory in which programs like *mount* search for certificates. When you attach to a server on another system the user name is passed across to the remote system as part of the attach procedure allowing the remote system to assign the correct ownership to files created on the remote server. The *wm* window manager program *wm/logon* allows a user to login to the local Inferno system before running the main window manager program *wm/wm*.

¡ **Step 11a: Re-start Inferno.**

You should now close down any instances of *emu* that you are currently running. The quickest way to do this is to type *control-c* in the emu window in which you ran *wm/wm*. Start a new *emu*, as before, by either running

```
emu
```

or by choosing the appropriate entry from your start menu on Windows machines. This time, run

```
wm/logon
```

and log in as user *inferno*. When you log in *wm/logon* will change directory to /usr/inferno and then write the name inferno to /dev/user. If this is the first time that you have logged on as user *inferno* to this machine it will display a licence. If the file /usr/inferno/namespace exists it will be used to construct a new namespace for the user based on the commands that it contains (see *newns*(2)) *Logon* will then will then start *wm/wm*.

### 12. What next

You should now have a fully functional Inferno system. You will need to have a three button mouse to use *acme*, *wm*, or *plumbing*.

To learn more you could start with the manual pages for: *intro*(1), *emu*(1), *wm*(1), *wm-misc*(1), *sh*(1), *acme*(1), and *limbo*(1) and also the papers in sections 1, 2 and 3 of volume two of the Inferno Programmer's Manual.