# 15 *General Troubleshooting*

One of the most challenging tasks performed by a system administrator is troubleshooting. Because troubleshooting is so important, it's a good idea to gain basic knowledge of some troubleshooting tools. This chapter describes several such tools. How to monitor and control system processesÐa frequent troubleshooting requirementÐis covered, as well as how to display host and network statistics, which can be used to identify network problems. Checking a disk for free space is also included. The remainder of the chapter describes procedures you need only in special situations, including repairing a NetInfo database and replacing a lost **root** password.

Keep in mind that each chapter in this manual contains troubleshooting tips. You find these tips in the last section of each chapter.

## Process Monitoring and Control

Each program that runs on the system is called a *process*. In case of trouble, you may need to examine the running processes and, in some cases, to destroy them. The Processes window (chosen from the Workspace Manager Tools menu) lets you monitor and destroy some processes. You can also use two UNIX commands: **ps**, to get detailed information about current processes, and **kill**, to destroy them.

A process is usually owned by the user who started it (although it can sometimes be owned by **root** or some other user). When a process is created, the program used to start that process (such as the Workspace Manager) is called its *parent process*. Similarly, the new process is called the *child* of that parent. In nearly all cases, a process has a parent and can create its own child processes. When using the **kill** command, this

process hierarchy can be important, as described later in this chapter.

# Examining Basic Process Information

The **ps** (process status) command lets you display a variety of information about the processes running on your system. Depending on the arguments you use when you execute **ps**, you can obtain a simple display or a highly complex one.

The simplest **ps** display lists the processes you own that are running on your terminal. The output of the **ps** command is similar to this:

```
PID TT STAT  TIME COMMAND
456 p1 S     0:01 -csh (csh)
```

The fields displayed are:

| Field | Description |
|---|---|
| PID | Process ID. Number uniquely identifying a process. |
| TT | Controlling terminal (or **tty**). Pseudoterminal to which the process is attached. |
| STAT | State of the process. Field made up of 4 characters (two or three will probably be blank) indicating such things as a sleeping process, a process that's swapped out, and so on. |
| TIME | Amount of CPU time used by the process. |
| COMMAND | Indication of the command being executed. In the previous example, **csh** is the C ShellÐthe command interpreter being run by the Terminal application. |

# Examining a Long Process Listing

If you want more detailed information about your processes, you can use the **-l** argument. The output of **ps -l** looks like this:

```
 F   UID  PID  PPID CP PRI BASE VSIZE RSIZE WCHAN STAT TT TIME COMMAND
201  201  141  140  0  10  10   1.26M  272K     0  S   p1 0:02 -csh (csh)
```

The additional fields displayed in this output are:

| Field | Description |
|---|---|
| F | Flags. A series of digits indicating flags that are associated with the process. For a list of flags, see the UNIX manual page for **ps**. |
| UID | User ID number associated with the process. |
| PPID | Parent process ID. Number that identifies the parent of the process. |
| CP | CPU utilization factor. |
| PRI | Scheduled priority level, derived from the base priority level. |
| BASE | Base priority level. Priorities are in the range 0-31; lower numbers mean lower priority. |
| VSIZE | Virtual size of the process (in kilobytes). |
| RSIZE | Real size of the process (in kilobytes). |
| WCHAN | Has no meaning on NeXT computers. |

# Examining Process Information for All Processes

Unless otherwise instructed, **ps** lists only processes that you've started in your current shell. To examine all the processes on the system, regardless of ownership, use the flags **ax**.

An example of the output from **ps -ax**:

```
PID TT STAT  TIME COMMAND
183 ?  S     1:26 /NextApps/Terminal.app/Terminal -MachLaunch 7 27
```

# Terminating a Process

Terminating a process is a simple way to correct certain problems. For example, if a shell window freezes, you might be able to free it by destroying the processes associated with it.

There may be times when you need to stop processes that can't be killed with the Workspace Manager Processes window. The **kill** command sends a signal to a specific process. There are a number of possible signals; several result in the process terminating.

To terminate a process with the **kill** command:

1. Start up the Terminal application, located in **/NextApps**.
2. Determine the process ID of the process you need to terminate by entering the following command:

   ```
   ps -ax
   ```

3. Make a note of the number in the PID column associated with the problem process.
4. Terminate the process with the following (replace *PID* with the number you noted in the previous step):

   ```
   kill PID
   ```

   **Note:** If you see an error message indicating that you are not the owner of the process, you'll need to use **su** to gain **root** access, then try again. You're only allowed to kill processes you own, but the superuser can kill just about any process.

5. Use the **ps** command to determine if the process is actually gone. If it's still around, try the following:

   ```
   kill -KILL PID
   ```

6. If the process won't die in response to **kill -KILL**, your only recourse is to reboot your computer.

# Guidelines for Destroying Processes

When determining how to destroy a process, here are some guidelines to follow:

1. Always try **kill** before using **kill -KILL**.

Using **kill -KILL** sends a signal that can't be ignored and should terminate any process. You should attempt a plain **kill** first rather than **kill -KILL** because the latter doesn't permit the process to clean up after itself and delete temporary files.

2. Always kill the children of a process first.

   Killing a process high in the hierarchy may halt more processes than necessary to solve the problem. In addition, one or more of the children processes may not exit, possibly compounding your problem.

3. Become **root** only if necessary.

   A mistake made as a regular user is rarely catastrophic; a mistake made as **root** can have very unfortunate consequences.

For more information, refer to the UNIX manual pages for **ps** and **kill**.

# Displaying Host and Network Statistics

Some utilities that display information that you may find useful for troubleshooting are: **hostinfo**, **ifconfig**, **netstat**, and **nidomain**. Each utility examines a different aspect of the network and then displays the resulting information. This section describes briefly how you might use each utility. For more detailed information, see the related UNIX manual pages.

## The hostinfo Command

The **hostinfo** utility gives you information about the NeXT computer you're logged into. After entering the command (which has no arguments), you see a screen display such as the following:

```
Mach kernel version:
  NeXT Mach 3.0: Mon Oct 8 18:52:05 PDT 1990;/ph1_sources/projects/mk-98/RELEASE
```

```
Kernel configured for a single processor only.
1 processor is physically available.
Processor type: MC680x0 (68040)
Processor active: 0
Primary memory available: 12.00 megabytes.
Default processor set: 44 tasks, 74 threads, 1 processors
Load average: 1.51, Mach factor: 0.09
```

This command makes it easy to find out how much memory you have available on your system. It also gives you a variety of information about the system processors, including how many processors were configured on the system, their logical slot numbers, and the CPU type.

# The ifconfig Command

The **ifconfig** utility can be used for many purposes, depending on the arguments used (for more details, see the UNIX manual page for **ifconfig**). For troubleshooting, you can use **ifconfig** with the *interface* argument to tell you how an Ethernet interface (or port) is configured and whether it's communicating with the network.

On NeXT computers, the Ethernet interface is always **en0**. Consequently, to get information about the Ethernet interface, enter the following:

```
ifconfig en0
```

You'll see a display similar to the following:

```
en0: flags=8023<UP,BROADCAST,NOTRAILERS>
  inet 192.42.172.13 netmask ffffff00 broadcast 192.42.172.255
```

This display tells you that the Internet address is 192.42.172.13, the netmask number is ffffff00, and the broadcast address is 192.42.172.255 (for more information about these values, see Appendix C, ªInternet Addressingº). The word ªUPº indicates that the interface can communicate with the network.

If the system can't transmit messages through the interface, the word ªUPº is omitted. You then see a display such as the following:

```
en0: flags=62<BROADCAST,NOTRAILERS>
        inet 0.0.0.0 netmask ff000000 broadcast 0.255.255.255
```

In this display, no Internet address is listed and the default broadcast address is used, indicating that no actual network was located.

# The netstat Command

The **netstat** utility displays information about network status. You can select from a variety of arguments that let you examine specific components of the network, such as the host table, the sockets, the routing tables, and so on. For a complete description of these arguments, see the UNIX manual page for **netstat**.

When used with the **-i** argument, **netstat** shows the state of the interfaces that have been autoconfigured. You see a display such as the following:

```
Name  Mtu   Network     Address      Ipkts  Ierrs    Opkts Oerrs  Coll
en0   1500  192.42.172  elmo        215694      0    80396     0     0
lo0   1536  loopback    localhost    18032      0    18032     0     0
en0*  1500  none        none        215920      0    80399     1     0
```

For debugging, you can examine the columns labeled **Ierrs** and **Oerrs**, which list the number of errors detected. Sometimes errors exist under normal conditions, so compare the number of errors you receive with other computers, and watch for more errors than usual on a specific machine. On older NeXT computers, the number of collisions reported (**Coll**) will always be zero.

# The nidomain Command

The command **nidomain -l** lets you find out information about the NetInfo domains served by a specified host (the **nidomain** utility can be used with other arguments; for a complete list, see the UNIX manual page for **nidomain**).

To get information about a particular host, enter the following (if you omit the host name, you receive information about the system you are currently logged into):

```
nidomain -l hostname
```

You see a display similar to the following:

```
tag=local udp=698 tcp=700
```

The **-l** argument displays the domains served by the specified host, along with the UDP port and TCP port used by **netinfod** for communication. Each domain is identified by its NetInfo tag. You might use this command if you suspect a corrupted NetInfo database. If a domain that should be listed is omitted, or an unexpected domain is included, a problem exists.

# Checking for Available Disk Space

If you want to load data onto your disk but aren't sure if there's room, or if you want to see whether a disk partition is getting too full, you can look at the File Viewer, or you can use the **df** command to check for free disk space in file systems. You can display the amount of used and available space, as well as how much of the total capacity of the file system has been used.

When **df** is executed without an argument, it reports on all mounted file systems. Here's an example of the output:

```
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/sd0a             345542    234167 76820    75%    /
```

You'll notice that the amount of used space (**used**) added to the available space (**avail**) is less than the amount of space in the file system (**kbytes**). This discrepancy arises because the system reserves a fraction of the space in the file system for use by its file system allocation routines. The amount reserved is usually around 10%, but you can adjust the amount reserved with the **tunefs** command. For further information, see the UNIX manual pages for **tunefs** and **df**.

You can check for space on a particular disk partition by using **df** with the partition as an argument. For example, the command **df** */dev/sd1a* results in output similar to the following:

```
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/sd1a             345542  241823   69164    78%    /Users
```

For additional information, see the UNIX manual page for **df**.

# Fixing NetInfo Problems

All NeXT computers use NetInfo to maintain administrative information. This section explains some steps you can take if, for some reason, NetInfo fails. It describes two utilities, **nidomain** and **niutil**, that let you access the NetInfo database. The **nidomain** command tells you what domains are served on a given machine, and lets you create and destroy NetInfo databases. The **niutil** command lets you read from and write to a particular NetInfo domain.

**Warning:** The debugging described in this section is for advanced administrators *only*. Errors made when working with the NetInfo database can have unforeseen results. This section assumes you're thoroughly familiar with UNIX and NetInfo. Before beginning, you should also read the UNIX manual pages for **niutil** and **nidomain**.

1. Turn the computer off and then on, using the Power key.
2. During the boot process, *immediately* after the ªTesting Systemº message is replaced by the ªLoading from diskº message, hold down the Command bar and press the ~ key (without pressing Shift). On keyboards with two Command keys, hold down the right Command key and press the ~ key. This displays the ROM monitor window containing the prompt NeXT>.
3. Start up the computer in single-user mode by entering one of the following commands at the ROM monitor prompt:

   | | |
   |---|---|
   | **`bsd -s`** | (To boot from the hard disk) |
   | **`bfd -s`** | (To boot from the internal floppy disk) |
   | **`bod -s`** | (To boot from the optical disk) |
   | **`ben -s`** | (To boot from the Ethernet) |

   When the startup process is complete, you'll see the single-user prompt #.
4. At the single-user prompt, start up the system services by entering the following command:

   ```
   sh /etc/rc &
   ```

   You see a series of messages appear on the screen as the **rc** shell script executes. During this process,

NetInfo is started. You won't see a shell prompt after these messages unless you press Return.

5. List the domains served by your system by entering the following:

   ```
   nidomain -l
   ```

   The system displays a list of existing domains, with one line per domain. If your system is not configured as a NetInfo server, you see just the local domain. If your system is a NetInfo server, you see all the domains served by this computer.

6. To make corrections in a specific domain, use the **niutil** command to examine and modify the NetInfo data. Always use the **-t** option with the appropriate tag identifier to avoid doing any lookups that would otherwise cause **niutil** to hang.

7. When you've fixed your data, restart the computer by powering it off and then on.

**Warning:** Don't try to continue the boot into multiuser mode. If you do, you'll start duplicate system services, which will cause a number of problems. You must halt the system and reboot.

Here's an example that shows how to display the properties associated with a specific NetInfo directory and to correct them if you find a problem. Suppose that someone mistakenly changed the user ID for **root** from 0 to 1. You first examine the NetInfo directory **/users/root** (located on the computer with the Internet address 127.0.0.1, in the NetInfo database tagged **local**) to see what properties it has:

```
niutil -read -t 127.0.0.1/local /users/root
name: root
passwd: e14vm5ROyurqA
uid: 1
gid: 1
realname: Operator
home: /
shell: /bin/csh
_writers_passwd: root
```

In this example, **name**, **passwd**, **uid**, **gid**, **realname**, **home**, **shell**, and **_writers_passwd** are properties of the directory **/users/root**. Once you see that the value of the **uid** property is 1 (an error), you can change it back to 0 with the following command:

```
niutil -createprop -t 127.0.0.1/local /users/root uid 0
```

For more information, see the UNIX manual pages for **nidomain** and **niutil**.

**Note:** You can't directly change a clone server database using this technique, since these are read-only. In other words, if the configured Internet address of the machine is not the same as that of the master server (identified in the **master** property of the NetInfo directory ª/º), this technique will not allow you to change the data. Instead, you can change the data on the master and then copy the database onto the clone using the command **nidomain -c**.

# Lost root Password

The **root** password is essential to system security and should always be safeguarded. Unfortunately, anything can be lost, and a password is no exception.

To assign a new **root** password when you're unable to log in as **root**, follow these procedures:

1. Turn the computer off and then on, using the Power key.
2. During the boot process, *immediately* after the ªTesting Systemº message is replaced by the ªLoading from diskº message, hold down the Command bar and press the ~ key (without pressing Shift). On keyboards with two Command keys, hold down the right Command key and press the ~ key. This displays the ROM monitor window.

**Warning:** If a hardware password has been set, you must supply this password before you can boot the computer. If you don't supply the hardware password, you'll be unable to complete the remainder of this procedure. (For more information on the hardware password, see ªThe ROM Monitorº in Chapter 9.)

3. Start up the machine in single-user mode by entering one of the following commands at the ROM monitor prompt:

| | |
|---|---|
| `bsd -s` | (To boot from the hard disk) |
| `bfd -s` | (To boot from the internal floppy disk) |
| `bod -s` | (To boot from the optical disk) |
| `ben -s` | (To boot from the Ethernet) |

The system starts up, displaying a series of messages. When startup is complete, you see the single user

prompt #.

4. At the single user prompt, start up the system services by entering the following command:

   ```
   sh /etc/rc &
   ```

   You see a series of messages appear on the screen as the **rc** shell script executes. During this process, NetInfo is started. You won't see a shell prompt after these messages unless you press Return.

5. Set the **root** password. Enter:

   ```
   passwd root
   ```

   You're prompted twice for the new password.

6. Turn the system off and then on again. After the system boots, you can resume normal operation with the new **root** password.

**Warning:** Don't try to continue the boot into multiuser mode. If you do, you'll start duplicate system services, which will cause a number of problems. You must halt the system and reboot.