General Approaches to Troubleshooting Troubleshooting is a complex process that is best learned through experience. This section looks briefly at how troubleshooting is done in order to see how these tools fit into the process. But while every problem is different - a key step is collecting information. Clearly - the best way to approach troubleshooting is to avoid it. If you never have problems - you will have nothing to correct. Sound engineering practices - redundancy - documentation - and training can help. But regardless of how well engi- neered your system is - things break. You can avoid troubleshooting - but you can't escape it. It may seem unnecessary to say - but go for the quick fixes first. As long as you don't fixate on them - they won't take long. Often the first thing to try is resetting the system. Many problems can be resolved in this way. Bit rot - cosmic rays - or the alignment of the planets may result in the system entering some strange state from which it can't exit. If the problem really is a fluke - resetting the system may resolve the problem - and you may never see it again. This may not seem very satisfying - but you can take your satisfaction in going home on time instead. Keep in mind that there are several different levels in resetting a system. For soft- ware - you can simply restart the program - or you may be able to send a signal to the program so that it reloads its initialization file. From your users' perspective - this is the least disruptive approach. Alternately - you might restart the operating system but without cycling the power - i.e. - do a warm reboot. Finally - you might try a cold reboot by cycling the power. You should be aware - however - that there can be some dangers in resetting a system. For example - it is possible to inadvertently make changes to a system so that it can't reboot. If you realize you have done this in time - you can correct the problem. Once you have shut down the system - it may be too late. If you don't have a backup boot disk - you will have to rebuild the system. These are - fortu- nately - rare circumstances and usually happen only when you have been making major changes to a system. When making changes to a system - remember that scheduled maintenance may involve restarting a system. You may want to test changes you have made - including their impact on a system reset - prior to such maintenance to ensure that there are no problems. Otherwise - the system may fail when restarted during the scheduled maintenance. If this happens - you will be faced with the difficult task of deciding which of several different changes are causing problems. Resetting the system is certainly worth trying once. Doing it more than once is a different matter. With some systems - this becomes a way of life. An operating   system that doesn't provide adequate memory protection will frequently become wedged so that rebooting is the only option.* Sometimes you may want to limp along resetting the system occasionally rather than dealing with the problem. In a university setting - this might get you through exam week to a time when you can be more relaxed in your efforts to correct the underlying problem. Or - if the system is to be replaced in the near future - the effort may not be justified. Usually - how- ever - when rebooting becomes a way of life - it is time for more decisive action. Swapping components and reinstalling software is often the next thing to try. If you have the spare components - this can often resolve problems immediately. Even if you don't have spares - switching components to see if the problem follows the equipment can be a simple first test. Reinstalling software can be much more prob- lematic. This can often result in configuration errors that will worsen problems. The old - installed version of

the software can make getting a new - clean installation impossible. But if the install is simple or you have a clear understanding of exactly how to configure the software - this can be a relatively quick fix. While these approaches often work - they aren't what we usually think of as troubleshooting. You certainly don't need the tools described in this book to do them. Once you have exhausted the quick solutions - it is time to get serious. First - you must understand the problem - if possible. Problems that are not understood are usually not fixed - just postponed. One standard admonition is to ask the question "has anything changed recently?" Overwhelmingly - most problems relate to changes to a working system. If you can temporarily change things back and the problem goes away - you have confirmed your diagnosis. Admittedly - this may not help with an installation where everything is new. But even a new installation can and should be grown. Pieces can be installed and tested. New pieces of equipment can then be added incrementally. When this approach is taken - the question of what has changed once again makes sense. Another admonition is to change only one thing at a time and then to test thoroughly after each change. This is certainly good advice when dealing with routine failures. But this approach will not apply if you are dealing with a system failure. (See the upcoming sidebar on system failures.) Also - if you do find something that you know is wrong but fixing it doesn't fix your problem - do you really want to change it back? In this case - it is often better to make a note of the additional changes you have made and then proceed with your troubleshooting. A key element to successful debugging is to control the focus of your investiga- tion so that you are really dealing with the problem. You can usually focus better

- Do you know what operating system I'm tactfully not naming? This is the Title of the Book
  - eMatter Edition Copyright 2007 O'Reilly & Associates
  - Inc. All rights reserved.

 if you can break the problem into pieces. Swapping components - as mentioned previously - is an example of this approach. This technique is known by several names—problem decomposition - divide and conquer - binary search - and so on. This approach is applicable to all kinds of troubleshooting. For example - when your car won't start - first decide whether you have an electrical or fuel supply problem. Then proceed accordingly. Chapter 12 outlines a series of specific steps you might want to consider. System Failures The troubleshooting I have described so far can be seen roughly as dealing with normal failures (although there may be nothing terribly normal about them). A second general class of problems is known as system failures. System failures are problems that stem from the interaction of the parts of a complex system in unexpected ways. They are most often seen when two or more sub- systems fail at about the same time and in ways that interact. However - system failures can result through interaction of subsystems without any ostensible failure in any of the subsystems. A classic example of a system failure can be seen in the movie China Syn- drome. In one scene the reactor scrams - the pumps shut down - and the water- level indicator on a strip-chart recorder sticks. The water level in the reactor becomes dangerously low due to the pump shutdown - but the problem is not recognized because the indicator gives misleading information. These two near-simultaneous failures conceal the true state of the reactor. System failures are most pernicious in systems with tight coupling between subsystems and subsystems that are linked in nonlinear or nonobvious ways. Debugging a system failure can be extremely difficult. Many of the more stan- dard approaches simply don't work. The strategy of

decomposing the system into subsystems becomes difficult - because the symptoms misdirect your efforts. Moreover - in extreme cases - each subsystem may be operating cor- rectly—the problem stems entirely from the unexpected interactions. If you suspect you have a system failure - the best approach - when feasible - is to substitute entire subsystems. Your goal should not be to look for a restored functioning system - but to look for changes in the symptoms. Such changes indicate that you may have found one of the subsystems involved. (Conversely - if you are working with a problem and the symptoms change when a sub- system is replaced - this is strong indication of a system failure.) Unfortunately - if the problem stems from unexpected interaction of nonfailing systems - even this approach will not work. These are extremely difficult prob- lems to diagnose. Each problem must be treated as a unique - special problem. But again - an important first step is collecting information.

Processes

- portmap
  - inetd
  - sendmail
  - telnetd
  - and chargen

Utilities

- `ps`
- `top` - useful for resource hogs
- `netstat`
  - reports the contents of kernel data structures related to networking
  - display the connections and services available on a host
  - list the routing table
  - modifying routing table
    - `ifconfig`
    - `route`
    - ICMP redirect
    - update from dynamic protocol like RIP or OSPF
- `lsof`
  - lists open files on a Unix system
  - -p - specify process number
  - -c - specify process name
  - -N - list open files for local computer on NSF server

    - -i - limit output to internet and X.25 network files
- `ifconfig`
    - alter the configuration of the network interfaces
- `arp`
    - allows examination or changing entries in the ARP table
    - ARP table on a system maps network addresses into MAC addresses
        - applies only to directly connected devices
        - remote devices
    - i.e.
    - devices that can be reached only by sending traffic through one or more routers
    - are not added to the ARP table since you can’t communicate with them directly. (However
    - the appropriate router interface will be added.)
        - addresses are added or removed automatically
    - for local system to communicate with another system on the local network whose MAC address is unknown
        - local system sends an ARP request
            - broadcast packet with the destination’s IP address
        - if the system is accessible
    - it will respond with an ARP reply that includes its MAC address
        - local system adds this to its ARP table and then uses this information to send packets directly to the destination
        - (can use ping)
    - most systems are configured to drop entries from the ARP table if they aren't being used
        - length of the timeout varies from system to system.
- `ip`
    - NOTE: replaces ifconfig and other networking commands on CentOS 7
    - etc

Port Scanners

- gtkportscan
    - nessus
    - portscan
    - and strobe
- nmap

Config Files

- start in /etc
    - /usr/local
    - /usr/opt
- files
    - defaultdomain
    - defaultroute
    - ethers
    - gateways
    - host.conf
    - hostname
    - hosts
    - hosts.allow
    - hosts.equiv
    - inetd.conf
    - localhosts
    - localnetworks
    - named.boot
    - netmasks
    - networks
    - nodename
    - nsswitch.conf
    - protocols
    - rc
    - rc.conf
    - rc.local
    - resolv.conf
    - services
- not all config files are on all systems, each version and release
- examine inetd.conf
    - compare with output of `netstat`
    - unmatched running proceses are cause for concern
- host.conf, resolv.conf, nsswitch.conf

- hosts.allow -> establishes trust relationships
- TODO: more here