Answers to the Assessment Test

1. B. The Monitor section defines the monitor options and settings but doesn't combine it with the video card, so option A is incorrect. The Modeline line defines the available video modes in the Monitor section, but it doesn't define video cards, so option C is incorrect. Option D, the Device section, is also incorrect; it defines the video card but doesn't match it with a monitor on the system. Option E is incorrect because the Module section defines which X server modules (or drivers) are loaded but it doesn't match monitors and video cards. Option B, the Screen section, tells the X server about the combination of video cards and monitors that you're using, so it's the correct answer. For more information, see Chapter 6, "Configuring the X Window System, Localization, and Printing."

2. A, C. Examining a process listing (obtained from `ps`) for signs of the super server is the most reliable way to determine which one is actually running, so option A is correct. The presence of the super server's configuration file or files (as in option C) is also a good diagnostic, although some older systems that have been upgraded may have both sets of configuration files. There is no standard `superserver` utility to report on which one is used, so option B is incorrect. Most distributions launch the super server through a SysV startup script; the `/etc/inittab` file isn't directly involved in this process, so examining it would be pointless, and option D is incorrect. Although the output of **`netstat -ap`**, when typed as `root`, will include an indication of any instance of `inetd` or `xinetd` that's listening for connections, option E omits the critical `-p` option, which causes the program to display process names. Thus, option E is incorrect. For more information, see Chapter 10, "Securing Your System."

3. D. The `lpc` utility is used to start, stop, change the priority of, and otherwise control jobs in a print queue. CUPS ships with an `lpc` utility, but it's quite rudimentary compared to the `lpc` utilities of BSD LPD and LPRng. Instead, CUPS relies on its Web-based interface to provide the ability to control print jobs. Thus, option D is correct, and the remaining options must logically all be incorrect. For more information, see Chapter 6.

4. C. The `/etc/security/limits.conf` file defines various limits on user resources, including the number of simultaneous logins individual users are permitted. Thus, option C is correct. The `/etc/pam.d/login-limits` file (option A) is fictitious, although login limits do rely on the `pam_limits` module to the Pluggable Authentication System (PAM). The `/etc/bashrc` file (option B) is a global `bash` startup script file, but it's not normally used to impose login limits. The `/etc/inittab` file (option D) is a key Linux startup file, but it doesn't have any direct bearing on imposing login limits. The `/etc/passwd` file (option E) defines many key account features, but login limits are not among these. For more information, see Chapter 10.

5. C, D. The computer's IP address (option C) and network mask (aka subnet mask or netmask; option D) are the most critical components in TCIP/IP network configuration. (Additional information that you may need to provide on many networks includes the IP addresses of one to three DNS servers, the hostname or IP address of a router, and the computer's hostname.) You shouldn't need the IP address of a Dynamic Host

Configuration Protocol (DHCP) server (option A)—and if a DHCP server is present, chances are you should be using DHCP rather than static IP address assignment. A NetBIOS Name Service (NBNS) server (option B) converts between names and IP addresses on NetBIOS networks. The hostname of such a computer isn't likely to be a critical configuration element, although you may need to provide this information to Samba for some operations to function correctly when sharing files. A Network Time Protocol (NTP) server (option E) helps you maintain system time on all of your computers, but this isn't required for basic network configuration. For more information, see Chapter 8, "Configuring Basic Networking."

6. E. The `wc` command displays a count of newlines, words, and bytes in the specified file (`report.txt`). Piping this data through `tee` causes a copy of the output to be stored in the new file (`wc` in this example—you shouldn't run this command in the same directory as the `wc` executable file!). Thus, option E is correct. Contrary to option A, `wc` is not an editor, and the remaining syntax wouldn't cause two files to open in separate windows even if `wc` were an editor. Contrary to option B, `wc` doesn't count windows or open a new window. Option C describes the effect of **`wc report | wc`**—that is, it overlooks the `tee` command. Contrary to option D, `wc` has nothing to do with cleaning up memory leaks, and `tee` doesn't directly use the `report.txt` file. For more information, see Chapter 1, "Exploring Linux Command-Line Tools."

7. C. The `grub.cfg` filename indicates a GRUB 2 configuration file. In such files, each OS or kernel stanza begins with a `menuentry` line and an open curly brace (`{`) and ends with a close curly brace (`}`). Thus, option C is correct. Some configuration files and programming languages use semicolons (`;`) at the end of most lines, but this isn't true of GRUB 2, so option A is incorrect. Although close parentheses (`)`) are used to terminate some types of options in some configuration files, including disk identifiers in GRUB 2's configuration file, they aren't used to terminate whole OS or kernel definitions in this file, so option B is incorrect. The string `*/` terminates comments in C program files but isn't commonly used in GRUB 2 configuration files, so option D is incorrect. Option E would be correct if the question had asked about a GRUB Legacy configuration file (`menu.lst` or `grub.conf`), but the question specifies a GRUB 2 configuration file (`grub.cfg`); the two boot loaders terminate their OS/kernel stanzas differently, so option E is incorrect. For more information, see Chapter 5, "Booting Linux and Editing Files."

8. E. The third field of `/etc/passwd` entries holds the UID number for the account, so option E is correct. Linux doesn't use any standard identifier called a human ID (HID; option A), although the acronym HID stands for human interface device, a class of USB devices. Accounts don't have PID numbers (option B); those belong to running processes. The account's GID number (option C) is stored in the fourth field of `/etc/passwd`—`100` in this example. Linux accounts don't use globally unique ID (GUID) numbers, so option D is incorrect. For more information, see Chapter 7, "Administering the System."

9. B. The `grep` command scans files to find those that contain a specified string or pattern, as described by option B. In the case of text files, `grep` displays the matching line or lines; for binary files, it reports that the file matches the pattern. The method of creating a pipeline (option A) involves separating two commands with a vertical bar (`|`). The `grep` command can be used in a pipeline, but it doesn't create one. The command that concatenates files (option C) is `cat`, and the command that displays the last several lines of a file (option D) is `tail`. Several commands, such as `find`, `locate`, and `whereis` locate files (option E), but `grep` is not among them. For more information, see Chapter 1.

10. B, D, E. ReiserFS (option B) was written from scratch for Linux. The Third Extended Filesystem (ext3fs; option D) is a journaling filesystem based on the older non-journaling Second Extended Filesystem (ext2fs; option C). The Extents Filesystem (XFS; option E) is a journaling filesystem written by SGI for Irix and later ported to Linux. The Virtual File Allocation Table (vfat; option A) is a non-journaling filesystem designed by Microsoft for Windows. For more information, see Chapter 3, "Configuring Hardware."

11. A. Option A correctly describes the features of SSH and GPG in this context. Option B is incorrect because SSH should do a fine job of encrypting your email so that it can't be decoded between your system and your ISP's email server. Option C has it backward; email transferred via SSH will be completely encrypted, including both headers and body. GPG doesn't encrypt headers, just message bodies. Option D is incorrect because GPG isn't a virus scanner, just an encryption tool. Option E is incorrect because the SSH tunnel will encrypt everything in the SMTP transfer, including email attachments. For more information, see Chapter 10.

12. A, D. Port 110 (option A) is assigned to the Post Office Protocol (POP), and port 143 (option D) is assigned to the Internet Message Access Protocol (IMAP), both of which may be used to retrieve email messages from an email server system. Port 119 (option B) is assigned to the Network News Transfer Protocol (NNTP), port 139 (option C) is assigned to the Server Message Block/Common Internet File System (SMB/CIFS) protocol, and port 443 (option E) is assigned to the Hypertext Transfer Protocol with SSL encryption (HTTPS), none of which is commonly used for email retrieval. For more information, see Chapter 8.

13. C. Log files, such as `/var/log/messages` and sometimes others in `/var/log`, often contain useful information concerning server errors. The `tail` program displays the last few lines of a file, so using it to examine log files immediately after a problem occurs can be a useful diagnostic procedure. Option C correctly combines these features. The `http://localhost:631` URL of option A accesses the Common Unix Printing System (CUPS) configuration utility, which has nothing to do with SSH. There is no standard `diagnose` utility (option B) to help diagnose server problems, and there is no standard `/dev/ssh` file (option D). The **`sshd`** program is the SSH server itself, so option E will simply launch the server. For more information, see Chapter 5.

14. B. The `~./profile` file is one of several `bash` startup scripts, as stated in option B. It has nothing to do with the ProFTP server (option A) or the `tcsh` shell (option D). The ProFile file manager mentioned in option C is fictitious. Users' encrypted passwords (option E) are usually stored in `/etc/shadow`. For more information, see Chapter 9, "Writing Scripts, Configuring Email, and Using Databases."

15. E. The `at` utility was created to run programs at one specified point in the future. Thus, option E will accomplish the stated goal. Options A and C might also work, but neither is the *best* way to accomplish this goal. Option A will tie up CPU time, and if the program

crashes or the system is shut down during the intervening two years, the message will never be displayed. Option C would be more reliable, but it adds unnecessary complexity to your hourly `cron` job schedule. The `cal` program displays a text-mode calendar, enabling you to identify the days of a week for a given month; it doesn't schedule future jobs, as option B suggests. A GUI calendar program, as specified in option D, might work, but NTP is the Network Time Protocol, a protocol and like-named program for synchronizing clocks across a network. Thus, NTP isn't the tool for the job, and option D is incorrect. For more information, see Chapter 7.

16. D. Option D provides the correct command to add 172.24.21.1 as the default gateway. Options A and B both use the fictitious `gateway` command, which doesn't exist and therefore won't work unless you create a script of this name. Option C uses the correct `route` command, but there is no `gateway` option to `route`; you must use `add default gw`, as in option D. There is no standard `gw` command, so option E is incorrect. For more information, see Chapter 8.

17. B, C. The BRLTTY package is an add-on daemon for handling a Braille display device, and some features for using these devices have been added to the 2.6.26 kernel, so options B and C are correct. Emacspeak (option A) is speech-synthesis software; it can be used to "speak" a text display to a user, but it doesn't interface with Braille displays. GOK (option D) is an onscreen keyboard, not a Braille display tool. Framebuffer drivers (option E) are kernel drivers for managing conventional video cards; they aren't used to drive Braille displays. For more information, see Chapter 6.

18. C. Some dependencies result from dynamically linking binaries to libraries at compile time, and so they can be overcome by recompiling the software from a source RPM, so option C is correct. Option A describes Debian source packages, not RPM packages. Recompiling a source RPM requires only issuing an appropriate command, although you must also have appropriate compilers and libraries installed. Thus, option B is overly pessimistic. Source tarballs can also be used to compile software for RPM systems, although this results in none of RPM's advantages. Thus, option D is overly restrictive. The RPM format doesn't impose any licensing requirements, contrary to option E. For more information, see Chapter 2.

19. D. The `mv` utility can be used to rename files as well as move them from one location to another, so option D is correct. The `dd` utility (option A) is used to copy files to backups, `rm` (option B) is used to remove (delete) files, `cp` (option C) copies files, and `ln` (option E) creates links. For more information, see Chapter 4.

20. B. Appending an ampersand (`&`) to a command causes that command to execute in the background. The program so launched still consumes CPU time, but it won't monopolize the shell you used to launch it. Thus, option B is correct. The `start` (option A) and `background` (option D) commands are fictitious. Although `bg` (option C) does place a job into the background, it doesn't launch a program that way; it places a process that has already been suspended (by pressing Ctrl+Z) into the background. The `nice` utility (option E) launches a program with modified priority, but a program so launched still monopolizes its shell unless you take additional steps. For more information, see Chapter 2.

21. A, B. The `-Uvh` parameter (option A) issues an upgrade command (which installs the program whether or not an earlier version is installed) and creates a series of hash marks to display the command's progress. The `-i` parameter (option B) installs the program if it's not already installed but causes no progress display. Option C uses a package name, not a complete filename, and so it will fail to install the package file. The `-e` option (option D) removes a package. Option E's `-Vp` option verifies the package file but doesn't install it. For more information, see Chapter 2.

22. B. Option B, `fsck`, is Linux's filesystem check utility. It's similar in purpose to the DOS and Windows `CHKDSK` and ScanDisk utilities (similar to options C and D), but these DOS and Windows utilities don't work on Linux filesystems like ext2fs or ReiserFS. Option A, `mkfs`, creates new filesystems; it doesn't diagnose or fix filesystem problems. Option E, `fdisk`, is a tool for creating or modifying disk partitions; it doesn't manage the filesystems they contain. For more information, see Chapter 3.

23. A. A freshly installed MySQL database is unlikely to have a ready-made database of animals, so your first task is to create that database with the `CREATE DATABASE` command, as shown in option A. (You could call the database something other than `animals`, of course.) The `USE` command in option B will be useful only once the database has been created. Once the database is created, you can use `CREATE TABLE`, as in option C, to create a table; however, you'll need an existing database first, and this command also requires information about the type of data to be stored, which option C doesn't provide. Option D's `INSERT INTO` command stores data into a table once it's been created, so it's far from the first command you'll use. It also requires additional specification of the data to be stored, so it's incomplete. Option E's `UPDATE` command modifies existing entries, so you'll use this command only after you've created the database and added at least one animal to it. (Option E is also an incomplete command even then.) For more information, see Chapter 9.

24. B, D. The correct answers, `man` and `info` (options B and D), are two common Linux help packages. Although `?` (option C) is a common help command within certain interactive programs, it isn't a help command in `bash` or other common Linux shells. There is no common command called `manual` (option A), nor is `hint` (option E) a valid `bash` command or common program name. For more information, see Chapter 1.

25. A, C. Unix systems traditionally store time in UTC (aka Greenwich mean time), and Linux may do so as well. Thus, option A is correct. Most other *x*86 PC OSs traditionally store time as the local time, however, so Linux also supports this option and option C is also correct. Internet Time (option B) is an alternative to the 24-hour clock in which the day is broken into 1,000 "beats." Standard PC BIOSs don't support this time format. Likewise, a 12-hour clock isn't terribly useful to computers because it doesn't differentiate a.m. from p.m., making option D incorrect. Although the length of the Martian day is similar to that of Earth (24 hours and 37 minutes), those wanting to colonize Mars will have to wait for PC clocks to support setting time for the Red Planet; option E is incorrect. For more information, see Chapter 7.

26. D. Typing **`lsmod`** (option D) produces a list of the modules that are currently loaded. The `insmod` (option A) and `modprobe` (option C) programs both load modules—either a single module or a single module and all those on which it depends, respectively. The `depmod` command (option B) generates the `modules.dep` file that contains module dependency information. The `modinfo` command (option E) displays information, such as its version number and author, on a single module. For more information, see Chapter 3.

27. B, E. The `chgrp` and `chown` commands can both change the group ownership of a file. The `chgrp` command takes a group name and a filename as parameters, as in option B. The `chown` command normally changes a file's owner; but if you provide a group name preceded by a dot (`.`) or a colon (`:`), as in option E, it changes the group of a file. The `chown` command as used in option A, will change the primary ownership of the file to the `music` user, if such a user exists on the system; it won't change the group ownership. There is no standard `chgroup` command, as in option C. Option D will change the permissions to 0600 (`-rw-------`), which will be a step backward with respect to the goal stated. For more information, see Chapter 4.

28. E. Hard links to directories are not permitted by most filesystems, so you'll probably have to create a symbolic link, as noted in option E. Links don't rely on a filesystem journal, so option A is incorrect. Contrary to option B, anybody may create a link, not just the original's owner. Option C describes a restriction of hard links, but because this link will probably have to be a symbolic link, this restriction is unimportant and option C is incorrect. Option D describes a more severe restriction than option B, but it's incorrect for the same reasons. For more information, see Chapter 4.

29. B, E. The colon (`:`) starts ex mode, from which you can enter commands. In ex mode, `r` includes a file in an existing one, `w` writes a file, `e` loads an entirely new file, and `q` quits the program. Thus the desired combination is `:wq` (option B). As a special case, `ZZ` does the same thing, so option E is also correct. For more information, see Chapter 5.

30. C. The `~/.forward` file is a user email forwarding file. The vertical bar character (`|`) at the start of such a file is a code to send the email through the specified program file, so option C is correct. To do as option A describes, the file would need to read `junkme` or `junkme@`*`hostname`*, where *`hostname`* is the computer's hostname. To do as option B describes, the leading vertical bar would have to be omitted. It's conceivable that the `~/junkme` script does as option D describes, but there's no way of knowing this for certain. To do as option E describes, the file would have to read *`user`*`@junkme`, where *`user`* is the username. For more information, see Chapter 9.

Chapter 1: Exploring Linux Command-Line Tools

1. D. Any of these approaches will work, or at least *might* work. (You might err when performing any of them.) Option B or C is likely to be the most efficient approach; with a long filename to type, option A is likely to be tedious.
2. E. The `echo` command is implemented internally to `bash`, although an external version is also available on most systems. The `cat`, `less`, `tee`, and `sed` commands are not implemented internally to `bash`, although they can be called from `bash` as external commands.
3. E. The `echo` command echoes what follows to standard output, and `$PROC` is an environment variable. Thus, `echo $PROC` displays the value of the `$PROC` environment variable, meaning that it must have been set to the specified value by you, one of your configuration files, or a program you've run. Although many environment variables are set to particular values to convey information, `$PROC` isn't a standard environment variable that might be associated with information described in options A, B, C, and D.
4. A. The `pwd` command prints (to standard output) the name of the current working directory. The remaining options are simply incorrect, although option B describes the `cd` command, and various tools can be used to reformat wide text for display or printing in fewer columns, as in option C.
5. A. The dot (`.`) character refers to the current working directory, and the slash (`/`) is a directory separator. Thus preceding a program name by `./` unambiguously identifies the intention to run the program that's stored in the current directory. Option B will run the first instance of the program that's found on the current path. Because paths often omit the current directory for security reasons, this option is likely to fail. The `run` command isn't a standard Linux command, so option C is unlikely to do anything, much less what the question specifies. Option D would be correct except that it reverses the order of the two characters. The effect is to attempt to run the `.myprog` file in the root (`/`) directory. This file probably doesn't exist, and even if it did, it's not the file the question specifies should be run. Option E runs the first instance of `myprog` found on the path, and additionally it runs the program in the background. (Chapter 2 covers background execution in more detail.)
6. E. By default, `man` uses the `less` pager to display information on most Linux systems, so option E is correct. Although an X-based version of `man` does exist (`xman`), the basic `man` doesn't use a custom X-based application (option A), nor does it use Firefox (option B) or the vi editor (option D). The `info` command and `man` are competing documentation systems, so option C is incorrect.
7. C. The `>` redirection operator stores a command's standard output in a file, overwriting the contents of any existing file by the specified name, so option C is correct.

Option A specifies the standard input redirection so that `ifconfig` will take the contents of `file.txt` as input. Option B is almost correct: the `>>` redirection operator redirects standard output, as requested, but it appends data to the specified file rather than overwriting it. Option D specifies a pipe; the output of `ifconfig` is sent through the `file.txt` program, if it exists. (Chances are it doesn't, so you'd get a `command not found` error message.) Option E redirects standard error, rather than standard output, to `file.txt` and so is incorrect.

8. C. The `&>` redirection operator sends both standard output and standard error to the specified file, as option C states. (The name of the file, `input.txt`, is intentionally deceptive, but the usage is still valid.) Option A mentions standard error but describes it as if it were an input stream, which it's not; it's an output stream. Option B mentions standard input, but the `&>` operator doesn't affect standard input. Because only option C is correct, neither option D nor E can be correct.

9. E. In principle, you can pipe together as many commands as you like. (In practice, of course, there will be limits based on input buffer size, memory, and so on, but these limits are far higher than the 2, 3, 4, or 16 commands specified in options A, B, C, and D.)

10. B. The `tee` command sends its output both to standard output and to a named file. Thus, placing the `tee` command (with an output filename) after another command and a pipe will achieve the desired effect. Options A and D redirect `gabby`'s output to a file, which means you won't be able to see the output and interact with it. Option C sends the contents of `gabby-out.txt` to `gabby` as input, which isn't what's desired, either. Option E attempts to run `gabby-out.txt` as a program and use its output as command-line arguments to `gabby`, which is not what's desired.

11. C. The `2>` redirection operator redirects standard error only, leaving standard output unaffected. Sending standard error to `/dev/null` gets rid of it. Thus option C is correct. Option A pipes the standard output of `verbose` through the `quiet` program, which isn't a standard Linux program. Option B sends both standard output and standard error to `/dev/null`, so you won't be able to interact with the program as the question specifies you must be able to do. Option D redirects standard output only to the `junk.txt` file, so once again, interaction will be impossible—and you'll see the unwanted error messages on the screen. Option E's `quiet-mode` program is fictitious (or at least nonstandard), so this option is incorrect.

12. A. Option A correctly describes the difference between these two redirection operators. Option B is almost correct, but the `>>` operator will create a new file if one doesn't already exist. The `>>` operator does not redirect standard error (as stated in option C) or standard input (as stated in option D). Both operators will create a new file if one doesn't already exist, contrary to what option E states.

13. C. The `tail` command displays the final 10 lines of a file, so option C is correct. (You can change the number of lines displayed with the `-n` option.) The `uniq` command (option A) removes duplicate lines from a list. The `cut` command (option B) echoes the specified characters or fields from an input text file. The `wc` command (option D)

displays counts of the number of characters, words, and lines in a file. The `fmt` command (option E) is a plain-text formatter.

14. A. The `pr` program takes a text file as input and adds formatting features intended for printing, such as a header and blank lines, to separate pages. The command also pipes the output through `lpr` (which is a Linux printing command). Option A describes these effects and so is correct. Option B describes the effect of the `cat` program and so is incorrect. The conversion of tabs to spaces can be done by the `expand` program, so option C is incorrect. Although the specified command does print `report.txt`, error messages are not stored in the `lpr` file, so option D is incorrect. Because option A is correct, option E is incorrect.
15. B, C, D. The `nl` command numbers lines, so it does this task without any special options, and option B is correct. (Its options can fine-tune the way it numbers lines, though.) The `cat` command can also number lines via its `-b` and `-n` options; `-b` numbers non-blank lines, whereas `-n` numbers all lines (including blank lines). Thus options C and D are both correct. Neither the `fmt` command nor the `od` command will number the lines of the input file, so options A and E are both incorrect.
16. D. The `expand` command will remove tab stops at every eight characters. With newly formatted data stored in `data1.txt` via the `>` redirection symbol, option D is the correct choice.

    The `od` command will not remove tabs. Therefore, option A is incorrect. Option B does remove the tabs; however, the resulting data file, `data·txt`, will contain duplicate data records (due to the `>>` redirection option), some with tabs and some without. Therefore, option B is incorrect. There is not a `--remove-tabs` option on the `fmt` command, and thus option C is incorrect. The `unexpand` command does the opposite of the `expand` command, adding tab stops instead of removing them. Therefore, option E is incorrect.
17. C. The `sed` utility can be used to "stream" text and change one value to another. In this case, the `s` option is used to replace `dog` with `mutt`, making option C correct. The syntax in option A is incorrect, and choices B and D are incorrect because `grep` doesn't include the functionality needed to make the changes. Option E combines `fmt`, `cut`, and redirection in a way that simply won't work to achieve the desired goal.
18. B. The `fmt` command performs the desired task of shortening long lines by inserting carriage returns. It sends its results to standard output, so option B uses output redirection to save the results in a new file. The `sed` command of option A won't accomplish anything useful; it only replaces the string `Ctrl-M` with the string `NL`. Although these strings are both sometimes used as abbreviations for carriage returns or new lines, the replacement of these literal strings isn't what's required. Option C creates an exact copy of the original file, with the long single-line paragraphs intact. Although option D's `pr` command is a formatting tool, it won't reformat individual paragraphs. It will also add headers that you probably don't want. Option E's `grep` command searches for text within files; it won't reformat text files.

19. A. The `grep` utility is used to find matching text within a file and print those lines. It accepts regular expressions, which means you can place in brackets the two characters that differ in the words for which you're looking. Thus option A is correct. The syntax for `sed`, `od`, `cat`, and `find` wouldn't perform the specified task, so options B through E are all incorrect.
20. C. The bracket expression within the `d[o-u]g` regular expression in option C means that any three-character string beginning in `d`, ending in `g`, and with the middle character being between `o` and `u` will match. These results meet the question's criteria. Option A's dot matches any single character, so `d.g` matches all three words. The bracket expression `[ou]` in option B matches the characters `o` and `u`, but no other values. Since the question specifies that some other matches will be made, this option is incorrect. Option D's `di*g` matches `dig`, `diig`, `diiig`, or any other word that begins with `d`, ends with `g`, and contains any number of `i` letters in between. Thus option D matches `dig` but not `dog` or `dug` as required. Option E, like option A, uses a dot to match any character, so it will actually match certain four-letter words but not `dog` or `dug`.

Chapter 2: Managing Software

1. D. Because they must be compiled prior to installation, source packages require *more* time to install than binary packages, contrary to option D's assertion, thus making this option correct. The other options all describe advantages of source packages over binary packages.
2. A. The two systems use different databases, which makes coordinating between them difficult. Therefore, using them both simultaneously is inadvisable, making option A correct. Package management systems don't share information, but neither do their databases actively conflict, so option B is incorrect. Installing the same libraries using both systems would almost guarantee that the files served by both systems would conflict with one another, making option C incorrect. Actively using both RPM and Debian packages isn't common on any distribution, although it's possible with all of them, so option D is incorrect. The `alien` program converts between package formats. Although it requires that both systems be installed to convert between them, `alien` is not required to install both these systems, thus option E is incorrect.
3. E. RPMs are usually portable across distributions, but occasionally they contain incompatibilities, so option E is correct. The package format and software licensing have nothing to do with one another, so option A is incorrect. There is no `--convert-distrib` parameter to `rpm`, so option B is incorrect. Although recompiling a source package can help work around incompatibilities, this step is not always required, so option C is incorrect. Binary packages can't be rebuilt for another CPU architecture, so option D is incorrect; although source packages may be rebuilt for any supported architecture, provided the source code doesn't rely on any CPU-specific features.

4. B. The `-i` operation installs software, so option B is correct. (The `-v` and `-h` options cause a status display of the progress of the operation, which wasn't mentioned in the option.) Uninstallation is performed by the `-e` operation, and rebuilding source RPMs is done by the `--rebuild` operation (to either `rpm` or `rpmbuild`, depending on the RPM version), so options A and C are incorrect. Although the filename `megaprog.rpm` is missing several conventional RPM filename components, the `rpm` utility doesn't use the filename as a package validity check, so option D is incorrect. Option E describes a package upgrade, which is handled by the `-U` operation, not `-i` as in the question, so option E is incorrect.
5. A. The `rpm2cpio` program extracts data from an RPM file and converts it into a `cpio` archive that's sent to standard output. Piping the results through `cpio` and using the `-i` and `--make-directories` options, as in option A, will extract those files to the current directory. Option B creates a `cpio` file called `make-directories` that contains the files from the RPM package. Option C will uninstall the package called `myfonts.rpm` (but not the `myfonts` package). The `alien` utility has no `--to-extract` target, so option D is invalid. The `rpmbuild` utility builds a source RPM into a binary RPM, making option E incorrect.
6. E. An uppercase `-P` invokes the purge operation, which completely removes a package and its configuration files, so option E is correct. The `-e` parameter uninstalls a package for `rpm`, but not for `dpkg`, so option A is incorrect. The lowercase `-p` causes `dpkg` to print information about the package's contents, so option B is incorrect. The `-r` parameter removes a package but leaves configuration files behind, so options C and D are both incorrect. (Option D also specifies a complete filename, which isn't used for removing a package—you should specify only the shorter package name.)
7. C. You can specify Debian package archive sites in `/etc/apt/sources.list`, and then you can type `apt-get update` and `apt-get upgrade` to update a Debian system quickly to the latest packages, so option C is correct. GUI package management tools for Debian and related distributions exist, but they aren't `apt-get`, so option A is incorrect. The `alien` program can convert a tarball and install the converted package on a Debian system, but `apt-get` can't do this, so option B is incorrect. `dpkg` and `apt-get` both come with all Debian-based distributions, so option D is incorrect. The `dpkg` program can install only Debian packages on Debian-based systems, but `apt-get` can work with both package systems, so option E is backward.
8. E. The `--get-selections` action to `dpkg` displays the names of all installed packages, making option E correct. There is no `showall` option to `apt-get`, so option A is incorrect. The `showpkg` subcommand to `apt-cache` displays information about a named package when used without a package name, as in option B, but it displays no data. The `dpkg -r` action removes a package, so option C would remove the package called `allpkgs` if it were installed. The `dpkg -i` action installs a package, so option D is incorrect—and that option doesn't list a package name, which the `-i` action requires.
9. D. The `update` option to `apt-get` causes retrieval of new information, as described in option D. This option is perfectly valid, contrary to option A's assertion.

The apt-get program doesn't permit you to upload information to the Internet repositories, so option B is incorrect. Option C describes the effect of the upgrade or dist-upgrade options, not the update option. The upgrade or dist-upgrade options can upgrade APT itself, but update alone won't do the job, so option E is incorrect.

10. A, B. The Yum utility's update and upgrade options are nearly identical in effect, and either can be used to upgrade an individual package, such as unzip, so options A and B are both correct. The primary command options to yum don't use dashes, so options C and D are both incorrect. The check-update option to yum checks for the availability of updates, but it does *not* install them, so option E is incorrect.
11. B. Yum uses files in the /etc/yum.repos.d directory to locate its repositories, so you can add to the repository list by adding files to this subdirectory, as option B specifies, typically either by installing an RPM or by adding a file manually. Option A describes a method of adding a repository to a computer that uses APT, not Yum. Option C's add-repository subcommand is fictitious. Although the /etc/yum.conf file described in options D and E is real, it doesn't store repository data.
12. B. The /etc/ld.so.conf file holds the global library path, so editing it is the preferred approach. You must then type ldconfig to have the system update its library path cache. Thus, option B is correct. Although you can add a directory to the library path by altering the LD_LIBRARY_PATH environment variable globally, as in option A, this approach isn't the preferred one, so this option is incorrect. Option C simply won't work. Option D also won't work, although linking individual library files would work. This method isn't the preferred one for adding a whole directory, though. The ldd utility displays information on libraries used by executable files, so option E won't have the desired effect.
13. D. Programmers select libraries, not users nor system administrators. If you don't like the widgets provided by one library, you have few options, and option D is correct. (Many widget sets do provide a great deal of configurability, though, so you may be able to work around the problem in other ways.) Options A, B, and E describe fictitious options to ldconfig, rpm, dpkg, and the kernel. Option C wouldn't work; Qt-using programs would crash when they found GTK+ libraries in place of the Qt libraries they were expecting.
14. D. The kill program accepts various signals in numeric or named form (9 in this example) along with a process ID number (11287 in this example). Signal 9 corresponds to SIGKILL, which is an extreme way to kill processes that have run out of control, thus option D describes the effect of this command. Although you might use kill to kill network processes, you can't pass kill a TCP port number and expect it to work, so option A is incorrect. The program also won't display information about the number of processes that have been killed, making option B incorrect. To do as option C suggests, you'd need to tell kill to pass SIGHUP (signal 1), so the command would be kill -1 11287, and option C is incorrect. The kill program can't change the priority of a process, so option E is incorrect.

15. C, D. The `top` utility displays a dynamic list of processes ordered according to their CPU use along with additional system information, including load averages, so option C is correct. If you want only the load average at a specific moment, `uptime` (option D) may be better because it presents less extraneous information—it shows the current time, the time since the system was booted, the number of active users, and the load averages. Option A's `ld` command has nothing to do with displaying load averages. (It's a programming tool that links together program modules into an executable program.) There are no standard Linux programs called `load` (option B) or `la` (option E).
16. A. The `--forest` option to `ps` shows parent-child relationships by creating visual links between process names in the `ps` output, making option A correct. (Listing 2.4 shows this effect.) Options B and C are both valid `ps` commands, but neither creates the specified effect. Option D describes a fictitious `ps` option. Since options B, C, and D are incorrect, option E is also necessarily incorrect.
17. A. CPU-intensive programs routinely consume 90 percent or more of available CPU time, but not all systems run such programs. Furthermore, some types of program bugs can create such CPU loads. Thus, option A is correct, and you must investigate the matter more. What is `dfcomp`? Is it designed as a CPU-intensive program? Is it consuming this much CPU time consistently, or was this a brief burst of activity? Options B, C, D, and E all jump to conclusions or present fictitious reasons for the behavior being normal or abnormal.
18. E. The `jobs` command summarizes processes that were launched from your current shell. When no such processes are running, `jobs` returns nothing, so option E is correct. The `jobs` command doesn't check or summarize CPU load, so option A is incorrect. The `jobs` command also doesn't check for processes run from shells other than the current one, so option B is incorrect (processes running under your username could have been launched from another shell or from a GUI environment). There is no standard `jobs` shell in Linux, so option C is incorrect. Because the `jobs` output is limited to your own processes in the shell you're running, a blank output does *not* indicate a crashed system, making option D incorrect.
19. C, E. The `nice` command launches a program (`crunch` in this example) with increased or decreased priority. The default priority when none is specified is 10, and the `nice -10 crunch` command also sets the priority to 10, so options C and E are equivalent. Option A isn't a valid `nice` command because `nice` has no `--value` option. Option B is a valid `nice` command, but it sets the priority to –10 rather than 10. Despite the similarity in the form of options C and D, option D is not a valid `nice` command, and so it is incorrect. (When passing a numeric value to `nice`, you *must* use a preceding dash, `-`, or `-n`.)
20. D, E. Linux insulates users' actions from one another, and this rule applies to `renice`; only `root` may modify the priority of other users' processes, so option D is correct. Similarly, only `root` may increase the priority of a process in order to prevent users

from setting their processes to maximum priority, thus stealing CPU time from others, so option E is correct. Option A correctly describes `nice`, but not `renice`. The whole point of `renice` is to be able to change the priorities of existing processes. Contrary to option B, `renice` doesn't care about the shell from which `renice` or the target program was launched. Users may use `renice` to decrease their own processes' priorities, contrary to option C.

Chapter 3: Configuring Hardware

1. **B, C.** IRQs 3 and 4 are common defaults for RS-232 serial ports, so options B and C are both correct. IRQ 1 is reserved for the keyboard, so option A is incorrect. IRQ 8 is reserved for use by the real-time clock, so option D is incorrect. Although IRQ 16 exists on modern systems, it didn't exist on early *x*86 systems, and its purpose isn't standardized.
2. **A.** Modern firmware (BIOSs and EFIs) provides the means to disable many onboard devices, including sound hardware, in case you don't want to use them, so option A is correct. Although the `alsactl` utility mentioned in option B is real, it's used to load or store sound card mixer settings, not to disable the sound hardware. The `lsmod` command mentioned in option C displays information about loaded kernel modules, but it doesn't remove them or disable the hardware they use. Similarly, option D's `lspci` displays information on PCI devices, but it can't disable them. Contrary to option E, on-board sound hardware can usually be disabled.
3. **E.** The `udev` software creates and manages a dynamic `/dev` directory tree, adding entries to that directory for devices that exist on the target system, so option E is correct. The `udev` software has nothing to do with software development (option A). It doesn't unload drivers (option B) or load drivers (option C), although it does respond to the loading of drivers by creating appropriate entries in `/dev`. It also doesn't store BIOS configuration options in a file (option D).
4. **E.** SATA disks are *usually* handled by Linux's SCSI subsystem and so are referred to as `/dev/sdx`. However, some drivers handle these disks as if they were PATA disks and so refer to them as `/dev/hdx`. Thus, option E is correct, and both options A and C are incorrect. The `/dev/mapper` directory holds device files related to LVM and RAID configurations, not disk partition identifiers, so option B is incorrect. Option D (`C:`) is how Windows would likely refer to the first partition on the disk, but Linux doesn't use this style of disk identifier.
5. **A, C, D.** There are no files called `/proc/ioaddresses` or `/proc/hardware`, so options B and E are both incorrect. All the other files listed contain useful information; `/proc/ioports` holds information about I/O ports, `/proc/dma` holds information about DMA port usage, and `/proc/interrupts` holds information about IRQs.

6. B. Logical partitions are numbered 5 and up, and they reside in an extended partition with a number between 1 and 4. Therefore, one of the first two partitions must be an extended partition that houses partitions 5 and 6, making option B correct. Because one of the first two partitions is an extended partition, the other must be a primary partition, and there can be no more of either type of partition. This makes option A incorrect. Gaps in the range of partitions 1–4 are normal in MBR disks, contrary to option C. Because logical partitions are numbered starting at 5, their numbers won't change if `/dev/sda3` is subsequently added, so option D is incorrect. On MBR disks, partitions 1–4 must be primary or extended partitions; logical partitions are numbered 5 and up. Thus option E is incorrect.
7. E . The `/etc/fstab` file contains the mapping of partitions to mount points, so `/etc` must be an ordinary directory on the root partition, not on a separate partition, making option E correct. Although option A's statement that the system won't boot is correct, the reason is not; `/home` holds user files, not critical system files. Options B and C describe restrictions that don't exist. Option D would be correct if `/etc` were not a separate partition.
8. D. The `/home` directory (option D) is frequently placed on its own partition in order to isolate it from the rest of the system and sometimes to enable use of a particular filesystem or filesystem mount options. The `/bin` and `/sbin` directories (options A and B) should *never* be split off from the root (`/`) filesystem because they contain critical executable files that must be accessible in order to do the most basic work, including mounting filesystems. The `/mnt` directory (option C) often contains subdirectories used for mounting removable media, or it may be used for this purpose itself. It's seldom used to access hard disk partitions directly, although it can be used for this purpose. The `/dev` directory (option E) usually corresponds to a virtual filesystem, which holds pseudo-files but is not stored on a disk partition.
9. A. The 0x0f partition type code is one of two common partition type codes for an extended partition. (The other is 0x05.) The 0x82 code refers to a Linux swap partition, and 0x83 denotes a Linux filesystem partition. Thus, it appears that this disk holds Linux partitions, making option A correct. Windows, FreeBSD, and Mac OS X all use other partition type codes for their partitions, so options B, C, and E are all incorrect. (Mac OS X is also rarely installed to MBR disks.) Partitions exist, in part, to enable different OSs to store their data side by side on the same disk, so mixing several partition types (even for different OSs) on one disk does not indicate disk corruption, making option D incorrect.
10. C. Linux's `fdisk` doesn't write changes to disk until you exit the program by typing **`w`**. Typing **`q`** exits without writing those changes, so typing **`q`** in this situation will avert disaster, making option C correct. Typing **`w`** (option B) would be precisely the wrong thing to do. Because `fdisk` doesn't write changes until you type **`w`**, the damage is not yet done, contrary to option A. Typing **`u`** (option D) or **`t`** (option E) would do nothing useful because those aren't undo commands.
11. E. The `mkfs` command creates a new filesystem, overwriting any existing data and therefore making existing files inaccessible, as stated in option E. This command

doesn't set the partition type code in the partition table, so option A is incorrect. The `mkfs` command is destructive, contrary to option B. The `-t ext2` option tells `mkfs` to create an ext2 filesystem; it's a perfectly valid option, so option C is incorrect. Although `mkfs` could (destructively) convert ext2fs to ext4fs, the `-t ext2` option clearly indicates that an ext2 filesystem is being created, so option D is incorrect.

12. B. Although they have similar names and purposes, Linux's `fdisk` isn't modeled after Windows's `FDISK`, so option B is correct and option A is not. Windows' `FDISK` does *not* have GUI controls, contrary to option C. Linux's `fdisk` does *not* format floppy disks, contrary to option D. Both programs manage MBR disks, contrary to option E.

13. E. Swap partitions aren't mounted in the same way as filesystems, so they have no associated mount points, making option E correct.

14. C. The `-t` option is used to tell `fsck` what filesystem to use, so option C is correct. (If this option isn't used, `fsck` determines the filesystem type automatically.) The `-A` option (option A) causes `fsck` to check all of the filesystems marked to be checked in `/etc/fstab`. The `-N` option (option B) tells `fsck` to take no action and to display what it would normally do without doing it. The `-C` option (option D) displays a text-mode progress indicator of the check process. The `-f` option (option E) is fictitious.

15. A. A default use of `df` reports the percentage of disk space used (option D) and the mount point for each filesystem (option E). The number of inodes (option B) and filesystem types (option C) can both be obtained by passing parameters to `df`. This utility does *not* report how long a filesystem has been mounted (option A), so that option is correct.

16. D. The journal of a journaling filesystem records pending operations, resulting in quicker disk checks after an uncontrolled shutdown, so option D is correct. Contrary to option A, journaling filesystems are, as a class, newer than non-journaling filesystems; in fact, the journaling ext3fs is built upon the non-journaling ext2fs. Although disk checks are quicker with journaling filesystems than with non-journaling filesystems, journaling filesystems do have `fsck` utilities, and these may still need to be run from time to time, so option B is incorrect. All Linux-native filesystems support Linux ownership and permissions; this isn't an advantage of journaling filesystems, contrary to option C. The journal of a journaling filesystem doesn't provide an unlimited "undo" feature, so option E is incorrect.

17. E. When typed without a filesystem type specification, `mount` attempts to auto-detect the filesystem type. If the media contains any of the specified filesystems, it should be detected and the disk mounted, so option E is correct.

18. B. The `/etc/fstab` file consists of lines that contain the device identifier, the mount point, the filesystem type code, filesystem mount options, the `dump` flag, and the filesystem check frequency, in that order. Option B provides this information in the correct order, and so it will work. Option A reverses the second and third fields, but is otherwise correct. Options C, D, and E all scramble the order of the first three fields and also specify the `noauto` mount option, which causes the filesystem not to mount automatically at boot time.

19. A, B, C. The `user`, `users`, and `owner` options in `/etc/fstab` all enable ordinary users to mount a filesystem, but with slightly different implications: `user` enables anybody to mount a filesystem—and only that user may unmount it; `users` enables anybody to mount a filesystem, and anybody may unmount it; and `owner` enables only the owner of the mount point to mount or unmount a filesystem. Thus, options A, B, and C are all correct. The `owners` parameter of option D doesn't exist. The `uid=1000` parameter of option E tells Linux to set the ownership of files to UID 1000 on filesystems that lack Linux permissions features. Although this might be desirable for some disks, it doesn't enable the user with UID 1000 to mount the disk, so option E is incorrect.
20. A. Option A correctly describes the safe procedure for removing a removable medium that lacks a locking mechanism from a Linux computer. (Instead of typing **`umount /media/usb`**, you could type **`umount /dev/sdb1`**; in this context, the two commands are equivalent.) Option B reverses the order of operations; the `umount` command *must* be typed *before* you physically remove the flash drive. Option C also has it backward; the `sync` command would need to be issued *before* removing the drive. (The `sync` command can prevent damage when removing disks, but it isn't a complete substitute for `umount`.) There is no standard `usbdrive-remove` command in Linux, and if you were to write a script that calls `umount` and call it `usbdrive-remove`, pulling the flash drive quickly, as option D describes, would be exactly the wrong thing to do. The `fsck` command of option E checks a filesystem for errors. It's not necessary to do this before removing a disk, and it won't unmount the disk, so option E is incorrect.

Chapter 4: Managing Files

1. B. The `touch` utility updates a file's time stamps, as option B specifies. (If the specified file doesn't exist, `touch` creates an empty file.) You can't move files with `touch`; that's the job of the `mv` command, so option A is incorrect. Various tools can convert end-of-line formats, but `touch` is not one of them, so option C is incorrect. Testing the validity of disk structures, as in option D, is normally done on a whole-filesystem basis with `fsck` and related tools; `touch` can't do this job. You can write cached data to disk for a whole filesystem by unmounting it or by using `sync`, but `touch` can't do this, so option E is incorrect.
2. A, D. The `-s` and `--symbolic` options to `ln` are equivalent, and both create a symbolic (aka soft) link. Thus, options A and D are both correct. Options B, C, and E don't exist.
3. A. The `-l` parameter produces a long listing, including file sizes. The `-a` parameter produces a listing of all files in a directory, including the dot files. Combining the two produces the desired information (along with information about other files), so option A is correct. The `-p`, `-R`, `-d`, and `-F` options don't have the specified effects, so the remaining options are all incorrect.

4. D. When moving from one partition or disk to another, `mv` must necessarily read and copy the file and then delete the original if that copy was successful, as stated in option D. If both filesystems support ownership and permissions, they'll be preserved; `mv` doesn't need an explicit `--preserve` option to do this, and this preservation does not rely on having exactly the same filesystem types. Thus, option A is incorrect. Although `mv` doesn't physically rewrite data when moving within a single low-level filesystem, this approach can't work when you're copying to a separate low-level filesystem (such as from a hard disk to a USB flash drive); if the data isn't written to the new location, it won't be accessible should the disk be inserted in another computer. Thus, option B is incorrect. Although not all filesystems support ownership and permissions, many do, and these attributes are preserved when moving files between them, so option C is incorrect. Although FAT is a common choice on removable media because of its excellent cross-platform support, other filesystems will work on such disks, so option E is incorrect.
5. A, B. If you try to create a directory inside a directory that doesn't exist, `mkdir` responds with a `No such file or directory` error. The `--parents` parameter tells `mkdir` to create all necessary parent directories automatically in such situations, so option A is correct. You can also manually do this by creating each necessary directory separately, so option B is also correct. (It's possible that `mkdir one` wouldn't be necessary in this example if the directory `one` already existed. No harm will come from trying to create a directory that already exists, although `mkdir` will return a `File exists` error.) Typing **`touch /bin/mkdir`**, as option C suggests, will likely result in an error message if typed as a normal user and won't help if typed as `root`, so this option is incorrect. Clearing away existing directories in the `one/two/three` tree won't help, so option D is incorrect. Option E's mktree command is fictitious.
6. D, E. The `cpio` and `tar` programs are common Linux archive-creation utilities, so options D and E are both correct. The `restore` command restores (but does not back up) data; its backup counterpart command is `dump`. Thus, option A is incorrect. The `vi` command launches a text editor; it's not used to create archives, so option B is incorrect. There is no standard `tape` command in Linux, so option C is incorrect.
7. E. With the `tar` utility, the `--list` (`t`) command is used to read the archive and display its contents. The `--verbose` (`v`) option creates a verbose file listing, and `--file` (`f`) specifies the filename—`data79.tar` in this case. Option E uses all of these features. Options A, B, C, and D all substitute other commands for `--list`, which is required by the question.
8. A. Symbolic links can point across filesystems, so creating a symbolic link from one filesystem (in which your home directory resides) to another (on the DVD) isn't a problem, making option A correct. Hard links, as in options B, C, and D, are restricted to a single filesystem and so won't work for the described purpose. Because symbolic links will work as described, option E is incorrect.
9. E. Option E is the correct command. Typing **`chown ralph:tony somefile.txt`**, as in option A, sets the owner of the file to `ralph` and the group to `tony`. The `chmod`

command used in options B and D is used to change file permissions, not ownership. Option C reverses the order of the filename and the owner.

10. C, E. The `d` character that leads the mode indicates that the file is actually a directory (option C), and the `r` symbol in the `r-x` triplet at the end of the symbolic mode indicates that all users of the system have read access to the directory (option E). Leading `l` characters, which this mode lacks, denote a symbolic link, so option A is incorrect. Although the `x` symbols usually denote executable program files, as specified in option B, in the case of directories this permission bit indicates that the directory's contents may be searched; executing a directory is meaningless. SUID bits are indicated by an `s` character in place of the owner's execute bit position in the symbolic mode. Since this position holds an `x` in this example, option D is incorrect.

11. C. The set user ID (SUID) bit enables programs to run as the program's owner rather than as the user who ran them. This makes SUID `root` programs risky, so setting the SUID bit on `root`-owned programs should be done only when it's required for the program's normal functioning, as stated in option C. This should certainly *not* be done for all programs because the SUID bit is *not* required of all executable programs, as option A asserts. Although the SUID `root` configuration does enable programs to access device files, the device files' permissions can be modified to give programs access to those files, if this is required, so option B is incorrect. Although SUID `root` programs are a security risk, as stated in option D, they're a necessary risk for a few programs, so option D goes too far. Many program files that should *not* be SUID `root` are owned by `root`, so option E is incorrect.

12. E. Using symbolic modes, the `o+r` option adds read (`r`) permissions to the world (`o`). Thus, option E is correct. Option A sets the mode to `rwxr----x`, which is a bit odd and doesn't provide world read access to the file, although it does provide world execute access. Option B sets the mode to `rw-r-----`, which gives the world no access whatsoever to the file. Option C adds read access to the file for the owner (`u`) if the owner doesn't already have this access; it doesn't affect the world permissions. Option D *removes* read access for *all* users, so it's incorrect.

13. D. Files start with a `666` permission bit octal number setting. Depending upon the umask setting, permission bits may be removed, but not added. Option D, `027`, removes write permissions for the group and all world permissions. (Files normally don't have execute permissions set, but explicitly removing write permissions when removing read permissions ensures reasonable behavior for directories.) Therefore, Option D is correct. Option A, `640`, is the octal equivalent of the desired `rw-r-----` permissions, but the umask sets the bits that are to be *removed* from permissions, not those that are to be set. Option B, `210`, would remove write permission for the owner, but it wouldn't remove write permission for the group, which is incorrect. This would also leave all world permissions open. Option C, `022`, wouldn't remove world read permission. Option E, `138`, is an invalid umask because all the digits in the umask must be between 0 and `7`.

14. E. Using quotas requires kernel support, the `usrquota` or `grpquota` (for user or group quotas) filesystem mount option, and activation via the `quotaon` command (which often appears in system startup scripts). Thus, option E is correct. Option A suggests that `quotaon` is not necessary, which is incorrect. Option B's statement that `grpquota` is invalid is incorrect. Option C's statement that these options *disable* quota support is backward. The `usrquota` and `grpquota` options are both valid, so option D is incorrect.
15. B. The `repquota` utility is used to summarize the quota information about the filesystem. When used with the `-a` option, it shows this information for all filesystems, so option B is correct. This command won't return useful information when typed alone, though, so option A is incorrect. The `quotacheck` utility checks quota information about a disk and writes corrections, so options C and D are both incorrect. The `edquota` utility enables you to edit quota information. It doesn't summarize quota information, and `-a` isn't a valid option to `edquota`. Thus, option E is incorrect.
16. D. The `/opt` directory tree exists to hold programs that aren't a standard part of a Linux distribution, such as commercial programs. These programs should install in their own directories under `/opt`; these directories usually have `bin` subdirectories of their own, although this isn't required. Thus, option D is correct (that is, it's a plausible possibility). The `/usr/sbin` directory holds programs that are normally run only by the system administrator, so it's not a likely location, making option A incorrect. The `/etc/X11` directory holds X-related configuration files; so it's very unlikely that WonderCalc will be housed there, making option B incorrect. The `/boot` directory holds critical system boot files, so option C is incorrect. The `/sbin` directory, like `/usr/sbin`, is an unlikely location for user files, so option E is incorrect. (Furthermore, `/sbin` seldom contains subdirectories.)
17. A. The `find` utility (option A) operates by searching all files in a directory tree, and so it's likely to take a long time to search all of a computer's directories. The `locate` program uses a precompiled database, `whereis` searches a limited set of directories, and `type` searches the shell's path and built-in commands, so these commands will take less time. Thus, options B, C, D, and E are all incorrect.
18. C. The `type` command identifies a command, as executed by the shell, as being a built-in shell command, a shell alias, or an external command, whereas the `whereis` command helps find the location of external command files, thus option C is correct. Neither `type` nor `whereis` identifies the CPU architecture of a program file, can locate commands based on intended purpose, complete an incompletely typed command, or identify a command as a binary or a script; thus, the remaining options are all incorrect.
19. B. The `find` command includes the ability to search by username using the `-user` *`name`* option, where *`name`* is the username; thus option B is correct. The `-uid` option to `find` can also locate files owned by a user, but it takes a numeric user ID (UID) as an argument, so option A isn't quite correct. The `locate` command provides no ability

to search by user, so options C and D are incorrect. Although option E is a valid `find` command, it finds all of the files under `/home` with a *filename* of `karen`, not all files owned by the user `karen`, so this option is incorrect.

20. D. The `which` program searches the path just as `bash` does, but it prints the path to the first executable program it finds on the path. Thus option D is correct. The `which` program doesn't conduct an exhaustive search of the system, so there could be many more files called `man` on the system, contrary to option A. System package tools and `which` aren't closely related; option B is incorrect. Although `/usr/bin/man` would be run when the user whose `which` output matches that in the question types **`man`**, this may not be true of others because the path can vary from one user to another, thus option C is incorrect. The `which` program doesn't reveal file ownership information, so option E is incorrect.

Chapter 5: Booting Linux and Editing Files

1. C. The Master Boot Record (MBR) can contain a boot loader that is up to 446 bytes in size, so option C is correct. If more space is required, the boot loader must load a secondary boot loader. Although the boot loader is loaded into RAM (option A), it's not stored there permanently because RAM is volatile storage. Both `/dev/boot` and `/dev/kmem` (options B and D) are references to files on Linux filesystems; they're only available after the system starts and lots of other boot processes have occurred. The swap partition (option E) is used as an adjunct to RAM; the BIOS won't look there for a boot loader.
2. C. Runlevel 1 is single-user mode, and adding the digit 1 to the kernel's options line in a boot loader will launch the system in this runlevel, so option C is correct. Options A and B both present invalid kernel options and so are incorrect. Although the `telinit` command specified in options D and E will change the runlevel once the computer is running and runlevel 1 is a single-user mode, these commands are *not* passed to the kernel via a boot loader, so these options are both incorrect.
3. D. The kernel ring buffer, which can be viewed by typing **`dmesg`** (piping this through `less` is a good supplement), contains messages from the kernel, including those from hardware drivers. These messages may provide a clue about why the disk didn't appear, thus option D is correct. The `/var/log/diskerror` file (option A) is fictitious, as is `/mnt/disks` (option B). The `/etc/inittab` file (option C) doesn't directly control disk access, and so it is unlikely to provide useful information. The files specified in option E are GRUB Legacy and GRUB 2 configuration files, which don't contain information that could explain why a disk isn't responding.

4. B. Ordinarily, Linux runs `init` (option B) as the first program; `init` then runs, via various scripts, other programs. The `dmesg` program (option A) is a user diagnostic and information tool used to access the kernel ring buffer; it's not part of the startup process. The `startup` program (option C) is fictitious. The `rc` program (option D) is a script that some versions of `init` call, typically indirectly, during the startup sequence, but it's not the first program that the kernel runs. LILO is an older boot loader for Linux on BIOS systems, and `lilo` (option E) is the command that installs this boot loader to the MBR. Since boot loaders run before the kernel loads, this option is incorrect.
5. D. Option D is the correct GRUB 2 configuration file. Option A is a fictitious file; it doesn't exist. Although some of GRUB 2's boot loader code may be written to the MBR, as implied by option B, this isn't the location of the program's configuration file. Options C and D are both possible names for the GRUB Legacy configuration file, but that name is not shared by GRUB 2.
6. A. The `initrd` keyword identifies an initial RAM disk file in the GRUB 2 configuration file, and a space separates this keyword from the filename. (Several variants on this syntax are possible.) Option B adds an equal sign (`=`), which renders the syntax incorrect. Options C, D, and E use the incorrect `initramfs` and `ramdisk` keywords instead of `initrd`.
7. D. You use `grub-install` to install the GRUB Legacy boot loader code into an MBR or boot sector. When using `grub-install`, you specify the boot sector on the command line. The MBR is the first sector on a hard drive, so you give it the Linux device identifier for the entire hard disk, `/dev/sda`. Hence, option D is correct. Option A specifies using the `grub` utility, which is an interactive tool, and the device identifier shown in option A is a GRUB-style identifier for what would probably be the `/dev/sda3` partition in Linux. Option B is almost correct, but it installs GRUB to the `/dev/sda1` partition's boot sector rather than to the hard disk's MBR. Option C is the command to install LILO to the MBR rather than to install GRUB. Option E contains the same error as option B, and it also uses the fictitious `grub-legacy` command.
8. B. The `root` keyword in a GRUB Legacy configuration file tells the boot loader where to look for files, including its own configuration files, kernel files, and so on. Because GRUB Legacy numbers both disks and partitions starting from 0, `(hd1,5)` refers to the sixth partition on the second disk, as option B specifies. Option A is incorrect because you pass the Linux root partition to the kernel on the kernel line, not via the GRUB `root` keyword. Options A, C, and E all misinterpret the GRUB numbering scheme. The GRUB installation location is specified on the `grub-install` command line, so options D and E are incorrect, and `/dev/hd1,5` isn't a standard Linux device file, which also makes option D incorrect.
9. B. The `initdefault` action specifies the default runlevel, so option B is correct. The remaining options are all taken from actual `/etc/inittab` files but don't have the specified meaning.

10. A, B, E. Runlevel 0 (option A) is the reserved runlevel for halting the system. Runlevel 1 (option B) is reserved for single-user mode. Runlevel 6 (option E) is reserved for rebooting. Runlevel 2 (option C) is the default runlevel on Debian and most distributions derived from it, but it does none of the things described in the question. Runlevel 5 (option D) is a regular, user-configurable runlevel, which isn't normally used for the things described in the question. (Many systems use it for a regular boot with a GUI login prompt.)
11. B, C. The first number in the `runlevel` output is the previous runlevel (the letter `N` is used to indicate that the system hasn't changed runlevels since booting). The second number is the current runlevel. Hence, options B and C are both correct, while options A and D are both incorrect. The runlevel changes very quickly, and the `runlevel` utility doesn't provide a code to indicate that the runlevel is in the process of being changed, so option E is incorrect.
12. A. The `-c` option to `shutdown` cancels a previously scheduled shutdown, as stated in option A. Options B and C describe the effects of the `-r` and `-h` options to `shutdown`, respectively. No `shutdown` option asks for confirmation before taking action, although you can delay a shutdown by specifying a shutdown time in the future, so option D is incorrect. No `shutdown` option closes open windows in X, except as a consequence of shutting down, so option E is incorrect.
13. B. The `journalctl` program displays the systemd log file, so option A is incorrect. Options C and D are commands used for the SysV initialization process, and Option E is the systemd process command. Option B, `systemctl`, is the correct answer.
14. B. The `telinit` command is used to change runlevels; when it's passed the `1` parameter, as in option B, `telinit` changes to runlevel 1, which is single-user mode. The `runlevel` command (option A) displays the current runlevel but doesn't change runlevels. Although `telinit` can be used to shut down or reboot the computer, the `shutdown` command (option C) can't be used to change runlevels except to runlevel 0 or 6. There is no standard `single-user` command (option D). The `halt` command (option E), like `shutdown`, can't be used to change to single-user mode.
15. A. The `isolate` command for the `systemctl` program allows you to change the target of the system. The rescue target specifies single-user mode. Option B changes the system to the default target, which may or may not be single-user mode. The `journalctl` program in Option C displays the systemd log files. The systemd program isn't used to change targets (Option D), and the `start` command only starts a single unit, not a target (Option E).
16. A. In vi, `dd` is the command-mode command that deletes lines. Preceding this command by a number deletes that number of lines. Thus option A is correct. Although `yy` works similarly, it copies (yanks) text rather than deleting it, so option B is incorrect. Option C works in many more-modern text editors, but not in vi. Option D works in Emacs and similar text editors, but not in vi. Option E works in many GUI text editors, but not in vi.
17. D. The `:q!` vi command does as option D states. Options A and E are both simply incorrect. Option B would be correct if this command was typed while in vi's insert

mode, but the question specifies that command mode is in use. To achieve option C, the command would be **`:wq`**, not **`:q!`**.

18. E. Vi is included on Linux emergency systems, embedded systems, and other systems where space is at a premium because its executable is tiny. Emacs is, in contrast, a behemoth. Thus option E is correct. Contrary to option A, vi isn't an X-based program (although X-based vi variants are available); Emacs can be used in text mode or with X. Extended Binary Coded Decimal Interchange Code (EBCDIC) is an obscure 8-bit character encoding system used on some very old mainframe OSs. When run on Linux, vi doesn't use EBCDIC; furthermore, EBCDIC offers few or no advantages over the American Standard Code for Information Interchange (ASCII). Thus option B is incorrect. Vi's modes, referred to in option C, have nothing to do with non-English language support. Option D is backward; it's Emacs that includes a web browser, email client, and other add-ons.
19. A, B, C. Typing **`R`** (option A) in command mode enters insert mode with the system configured to overwrite existing text. Typing **`i`** or **`a`** (options B and C, respectively) enters insert mode with the system configured to insert text. (The `i` and `a` commands differ in how they place the cursor; `a` advances one space.) Typing **`:`** (option D) in command mode enters ex mode (you typically type the ex-mode command on the same command line immediately after the colon). Pressing the Esc key (option E) returns vi to command mode from insert mode.
20. B. The Esc key exits vi's insert mode, as option B specifies. Typing a tilde (~) inserts that character into the file, so option A is incorrect. The Ctrl+X, Ctrl+C key combination exits from Emacs, but it's not a defined vi key sequence, so option C is incorrect. The F10 key and the Shift+Insert key combination also aren't defined in vi, so options D and E are both incorrect.

Chapter 6: Configuring the X Window System, Localization, and Printing

1. A. On most Linux systems, some runlevels don't run X by default, so using one of them along with the `startx` program (which starts X running) can be an effective way to test changes to an X configuration quickly, making option A correct. The `telinit` program changes runlevels, which is a lengthy process compared to using `startx`, so option B is incorrect. Unplugging the computer to avoid the shutdown process is self-defeating because you'll have to suffer through a long startup (if you use a non-journaling filesystem), and it can also result in data loss, thus option C is incorrect. The `startx` utility doesn't check the veracity of an X configuration file; it starts X running from a text-mode login, making option D incorrect. Reconfiguring an X server does not normally require network access; the X server runs on the computer at which you sit. Thus option E is incorrect.
2. D. The `XF86Config` and `xorg.conf` file design enables you to define variants or multiple components and easily combine or recombine them as necessary, using the structure

specified in option D. Options A, B, and C all describe fictitious structures. Option E is incorrect because the X.org-X11 and XFree86 configuration files use a text-mode structure, not a binary structure.

3. C. The vertical refresh rate range includes a maximum value, but that value may be reduced when the resolution and vertical refresh rate would demand a higher horizontal refresh rate than the monitor can handle. Thus, option C is correct. Since the resolution affects the maximum refresh rate, option A is incorrect. The color depth is irrelevant to resolution and refresh rate calculations, so option B is incorrect. The computations shown in options D and E are bogus, making these options incorrect.
4. E. Option E describes the correct location for this option. The `ServerLayout` section (referenced in option A) combines all of the other options together but doesn't set the resolution. The `Modeline` option in the `Monitor` section (as described in option B) defines *one* possible resolution, but there may be several `Modeline` entries defining many resolutions, and there's no guarantee that any of them will be used. The `Modeline` option doesn't exist in the `Device` section (as suggested by option C), nor is that section where the resolution is set. There is no `DefaultResolution` section (as referenced in option D).
5. B. By maintaining fonts on one font server and pointing other X servers to that font server, you can reduce the administrative cost of maintaining the fonts on all of the systems, so option B is correct. Font servers don't produce faster font displays than X's local font handling; if anything, the opposite is true. Thus, option A is incorrect. XFree86 4.*x* supports TrueType fonts directly, so option C is incorrect. Converting a bitmapped display into ASCII text is a function of optical character recognition (OCR) software, not a font server, so option D is incorrect. Neither X core fonts nor a font server handles font smoothing; for that, you need Xft. Thus, option E is incorrect.
6. C, E. XDMCP servers are typically launched either from a system startup script or by `init` (as specified in `/etc/inittab`), as described in options C and E. The XDMCP server then starts X. The `Start` folder mentioned in option A is a Windows construct, not a Linux construct. The `~/.xinitrc` script mentioned in option B is an X login script used when starting X from the command line via `startx`; it's not used to start X automatically when the system boots. A boot manager, as described in option D, launches the kernel; it doesn't directly start X, so option D is incorrect.
7. E. The XDM greeting is a resource set in the `/etc/X11/xdm/Xresources` file, so option E is correct. XDM doesn't offer many options on its main screen and certainly not one to change its greeting, as described in option A. The kernel doesn't directly handle the login process, nor does it pass options directly to XDM, so option B is incorrect. Although the `xorg.conf` file mentioned in option C is real, this file provides no XDM configuration options because XDM is a separate program from the X server. There is no standard `xdmconfig` program, as mentioned in option D.
8. C. KDM and GDM add many features, one of which is a menu that enables users to select their desktop environment or window manager when they log in rather than specifying it in a configuration file, as option C states. Option A describes one of the

advantages of the Secure Shell (SSH) as a remote-access protocol. Option B describes a feature common to all three XDMCP servers. Option D describes the way both KDM and XDM function; GDM is the one that presents username and password fields in series rather than simultaneously. Although a failure of X to start usually results in a fallback to a text-mode login, this feature is not provided by the XDMCP server, so option E is incorrect.

9. A. The `xhost` command controls various aspects of the local X server, including the remote computers from which it will accept connections, making option A correct. Option B sets the `DISPLAY` environment variable, which doesn't directly affect the X server (it does tell X clients which X server to use). Option C initiates a text-mode remote login session with `penguin.example.com`. Option D's `xaccess` is a fictitious program. Although logging into penguin.example.com via `ssh` may also initiate an X tunnel, this isn't guaranteed, and such a tunnel doesn't cause the local X server to accept *direct* connections from the remote computer, so option E is incorrect.

10. A. As stated in option A, GNOME, KDE, and other user programs often override the keyboard repeat settings in the X configuration file. Option B has it almost backward; most Linux distributions have abandoned XFree86, and therefore its `XF86Config` file, in favor of X.org-X11 and its `xorg.conf` file. Option C is pure fiction; `xorg.conf` settings apply to all varieties of keyboards, and there is no standard `usbkbrate` program. Although some keyboards do have hardware switches, they don't affect X's ability to control the keyboard repeat rate, contrary to option D. Although you can set a keyboard's nationality in `xorg.conf`, this option is independent of the keyboard repeat rate settings, so option E is incorrect.

11. C, E. The Orca and Emacspeak programs both provide text-to-speech conversion facilities, so options C and E are both correct. Braille is a form of writing that uses bumps or holes in a surface that can be felt by the reader. Although Linux supports Braille output devices, the question specifies computer-generated speech, which Braille is not, so option B is incorrect. SoX (option A) is an audio format converter, but it won't convert from text to speech. The `talk` program (option D) is an early Unix online text-mode "chat" program, but it has no built-in speech synthesis capabilities.

12. B, E. Time zones are determined by the `/etc/localtime` file, so replacing that one with the correct file (a selection is stored in `/usr/share/zoneinfo`) will fix the problem, making option B correct. (You may also need to edit `/etc/timezone` or some other file to keep automatic utilities from becoming confused.) Utilities such as `tzselect` will make these changes for you after prompting you for your location, so option E is also correct. The `hwclock` program mentioned in option A reads and writes data from the system's hardware clock. Although it relies on time zone data, it can't adjust your system's time zone itself. There is no standard `/etc/tzconfig` file, although the `tzconfig` program, like `tzselect`, can help you set the time zone. Thus, option C is incorrect. The `/etc/localtime` file is a binary format; you shouldn't attempt to edit it in a text editor, making option D incorrect.

13. D. Linux, like Unix, maintains its time internally in Coordinated Universal Time (UTC), so setting the computer's hardware clock to UTC (option D) is the recommended procedure for computers that run only Linux. Although Linus Torvalds spent time at the University of Helsinki, Helsinki time (as in option A) has no special place in Linux. Local time (as in option B) is appropriate if the computer dual-boots to an OS, such as Windows, that requires the hardware clock to be set to local time, but this is the second-best option for a Linux-only system. Option C's US Pacific time, like Helsinki time, has no special significance in Linux. Internet time (option E) is an obscure way to measure time that divides each day into 1,000 "beats." It's not a time zone and is not an appropriate way to set your hardware clock.
14. C. When set, the `LC_ALL` environment variable (option C) adjusts all the locale (`LC_*`) variables, so setting this and then running the script will make the programs that your script uses work as if on a British computer. The BIOS has no location code data, so option A is incorrect. There is no standard `/etc/locale.conf` file, so option B is incorrect. There is no standard `locale_set` utility, so option D is incorrect. Although setting the `TZ` environment variable, as in option E, will set the time zone for your local shell to that for Great Britain, this won't affect the sort of text formatting options noted in the question.
15. A. The Unicode Transformation Format 8 (UTF-8) standard can encode characters for just about any language on Earth, while looking just like ordinary ASCII to programs that only understand ASCII. Thus UTF-8 (option A) is the preferred method for character encoding when a choice is possible. ASCII (option B) is an old standard that's adequate for English and a few other languages, but it lacks some or all characters needed by most languages. ISO-8859 (options C and D) is a standard that extends ASCII, but it requires separate encodings for different languages and so it is awkward when a computer must process data from multiple languages. ATASCII (option E) is a variant of ASCII used in the 1980s by Atari for its home computers; it's obsolete and inadequate today.
16. E. The smart filter makes a print queue "smart" in that it can accept different file types (plain text, PostScript, graphics, and so on) and print them all correctly, as in option E. Font smoothing is useful on low-resolution computer monitors, but not on most printers, and adding font smoothing is not a function of a smart filter, so option A is incorrect. A smart filter doesn't detect confidential information (option B) or prank print jobs (option D). The `lpr` program can be given a parameter to email a user when the job finishes (option C), but the smart filter doesn't do this.
17. B, D. The job ID (option B) and job owner (option D) are both displayed by `lpq`. Unless the application embeds its own name (option A) in the filename, that information won't be present. Most printers lack Linux utilities to query ink or toner status (option C); certainly `lpq` can't do this. Although knowing when your job will finish printing (option E) would be handy, providing this information is well beyond `lpq`'s capabilities.
18. C. The `lprm` command (option C) deletes a job from the print queue. It can take the `-P`*`queue`* option to specify the queue and a print job number or various other

parameters to specify which jobs to delete. BSD LPD, LPRng, and CUPS all implement the `lprm` command, so you can use it with any of these systems, making option A incorrect. Option B presents the correct syntax but the wrong command name; there is no standard `lpdel` command. The `cupsdisable` command can be used to disable the whole queue but not to delete a single print job, so option D is incorrect. Because option C is correct, option E obviously is not.

19. B. PostScript is the de facto printing standard for Unix and Linux programs, as specified in option B. Linux programs generally *do not* send data directly to the printer port (option A); on a multitasking, multiuser system, this would produce chaos because of competing print jobs. Although a few programs include printer driver collections, most forgo this in favor of generating PostScript, making option C incorrect. Printing utilities come standard with Linux; add-on commercial utilities aren't required, so option D is incorrect. Verdana is one of several "web fonts" released by Microsoft. Although many Linux programs *can* use Verdana for printing if the font is installed, most Linux distributions don't install Verdana by default, and few Linux programs use it for printing by default even if it's installed, so option E is not correct.

20. B. The `mpage` utility (option B) prints multiple input pages on a single output page, so it's ideally suited to the specified task. PAM (option A) is the Pluggable Authentication Modules, a tool for helping to authenticate users. 4Front (option C) is the name of a company that produces commercial sound drivers for Linux. The `route` command (option D) is used to display or configure a Linux routing table. The `411toppm` program (option E) converts files from Sony's 411 image file format to the PPM image file format; it doesn't do the specified task.

Chapter 7: Administering the System

1. E. When the `usermod -L` *`username`* command is used, the *`username`* record in the `/etc/shadow` file has its password field modified. An exclamation point (`!`) is placed in front of the password, making the password inoperable and thus locking the account. Therefore, option E is correct. An `x` exists in the `/etc/passwd` file's records' password field, if the `/etc/shadow` file is used for passwords (which it should be) and does not indicate a locked account. Therefore, option A is incorrect. Option B is only true when an account has not yet had a password set. Therefore, option B is incorrect. Option C is also incorrect. You would never have a blank password field for a user account's `/etc/shadow` record, unless the file had been incorrectly manually modified. Manual modifications of the `/etc/shadow` files are *never* recommended. A user record could have a zero (0) as the first character in their password field, but this would be due to the password being hashed, not locked. Therefore, option D is incorrect.

2. A, B, C. The `useradd` command is used to add user accounts to a Linux system, and therefore option A is correct. The `adduser` command is available on some Linux distributions, and it also allows you to add user accounts to the system. Thus, option B is correct as well. The `useradd` command has a valid `-c` option that allows you to enter comments, such as a user's full name. Therefore, option C is also correct. There is no `usradd` command, so option D is incorrect. The `passwd` command cannot add users to the system. Therefore, option E is incorrect.

3. A. The `chage` command changes various account expiration options. The `-M` parameter sets the maximum number of days for which a password is valid, and in the context of the given command, *`time`* is a username. Thus, option A is correct. Options B, C, D, and E are all made up.
4. D. The `/etc/passwd` entries have third and fourth fields of the UID and the GID, but this line has only one of those fields (which one is intended is impossible to determine); this example line's fourth field is clearly the fifth field of a valid entry. Thus, option D is the correct answer. Option A is incorrect because, although `/bin/passwd` is an unorthodox login shell, it's perfectly valid. This configuration might be used on, say, a Samba file server or a POP mail server to enable users to change their passwords via SSH without granting login shell access. The `sally` username is valid and thus, Option B is not a correct answer. You may have usernames that are all lowercase letters. Option C is a correct observation, but an incorrect answer; the username and the user's home directory name need not match. The hashed password is officially stored in the second field, but in practice, most Linux computers place the hashed passwords in the `/etc/shadow` file. An `x` value for the password is consistent with this use, so option E is incorrect.
5. E. Option E is the best way to accomplish the task, because it will add `sally` to the `Development` group without removing her from any other groups or potentially damaging the `/etc/group` file. Option A would attempt to add the groups `Development` and `sally` to the system, thus it is not even a valid choice. Option B, also not a valid choice, would attempt to add the groups `Production` and `sally`. Option C would work, but it is very dangerous to edit an account configuration file manually instead of using account tools. Therefore, option C is not the best choice. Option D would work, but it would remove `sally` from all of her other groups, including the `Production` group. Therefore, option D is not the best choice either.
6. B, C, D. Files in `/etc/skel` are copied from this directory to the new users' home directories by certain account-creation tools. Thus, files that you want in all new users' home directories should reside in `/etc/skel`. Options B, C, and D all describe reasonable possibilities, although none is absolutely required. Including a copy of `/etc/shadow` in `/etc/skel` (option A) would be a very bad idea because this would give all users access to all other users' hashed passwords, at least as of the moment of account creation. You wouldn't likely find package management databases (option E) in `/etc/skel`, since users don't need privileged access to this data, nor do they need individualized copies of it.
7. C. The `userdel` command deletes an account, and the `-r` option to `userdel` (option C) causes it to delete the user's home directory and mail spool, thus satisfying the terms of

the question. Option A deletes the account but leaves the user's home directory intact. Option B does the same; the `-f` option forces account deletion and file removal under some circumstances, but it's meaningful only when `-r` is also used. Option D's `rm` command deletes the user's home directory (assuming that it's located in the conventional place, given the username) but doesn't delete the user's account. Option E's `usermod` command can modify accounts, including locking them, but it can't delete accounts. Furthermore, the `-D` option to `usermod` is fictitious.

8. E. The `emerg` priority code (option E) is the highest code available and so is higher than all the other options. From highest to lowest priorities, the codes given as options are `emerg`, `crit`, `warning`, `info`, and `debug`.
9. A. The `logrotate` program consults a configuration file called `/etc/logrotate.conf` (option A), which includes several default settings and typically refers to files in `/etc/logrotate.d` to handle specific log files. The remaining options are all fictitious, at least as working log files for `logrotate`.
10. D. The `logger` utility can be used to create a one-time log file entry that you specify. In its simplest form, it takes no special arguments, just a message to be inserted in the log file, as in option D. The `dmesg` utility in option A is used to review the kernel ring buffer; it doesn't create log file entries. Option B's `syslog` command isn't a Linux user-mode command, although it is the name of the logging system generically as well as a programming language command name. Option C's `rsyslogd` is the name of one of several system logging daemons; it maintains the system log, but isn't used to manually insert log entries. Option E's `wall` command writes a message to all users logged into virtual console terminals. It won't create a log file entry as the question requires and is not installed on all distributions.
11. C. The `logrotate` program can be started automatically—and unattended—on a regular basis by adding an entry for it in `cron`, so option C is correct. The `at` utility (option A) would be used if you wanted the program to run only once. Option B, `logrotate.d`, is a file stored in the `/etc` directory, which defines how the program is to handle specific log files. The `inittab` file (option D) is used for services and startup and not for individual programs. The `ntpd` program (option E) is the Network Time Protocol daemon, which synchronizes the system's clock with outside time sources.
12. E. The `hwclock` utility is used to view or set the hardware clock. The `--systohc` sets the hardware clock based on the current value of the software clock, thus option E is correct. Option A's `date` utility can be used to set the software clock but not the hardware clock; it has no `--sethwclock` option. Option B's `ntpdate` is used to set the software clock to the time maintained by an NTP server; it doesn't directly set the hardware clock. Option C's `sysclock` utility is fictitious. Option D's `time` command is used to time how long a command takes to complete; it has no `--set` or `--hw` option and does not set the hardware clock.
13. A. The format of the `date` command's date code is `[`*MMDDhhmm*`[[`*CC*`]`*YY*`][.`*ss*`]]`. Given that the question specified an eight-digit code, this means that the ordering of the

items, in two-digit blocks, is month-day-hour-minute. Option A correctly parses this order, whereas options B, C, D, and E do not.

14. C. Multiple `server` entries in `/etc/ntp.conf` tell the system to poll all of the named servers and to use whichever one provides the best time data. Thus option C is correct. (The `pool.ntp.org` subdomain and numbered computers within that subdomain give round-robin access to a variety of public time servers.) Options A and B both incorrectly state that one `server` statement overrides another, when in fact this isn't the case. The `server` statements shown in the question are properly formed. These `server` entries are properly formed, so option D is incorrect. Although it is true that this configuration will result in use of `tardis.example.com` should the public-pool server be unavailable, as option E states, this is not the *only* reason the NTP server will use `tardis.example.com`; this could happen if the public-pool server provides an inferior time signal, for instance. Thus option E is incorrect.

15. D. Once you've configured one computer on your network to use an outside time source and run NTP, the rest of your computers should use the first computer as their time reference. This practice reduces the load on the external time servers as well as your own external network traffic. Thus option D is correct. (Very large networks might configure two or three internal time servers that refer to outside servers for redundancy, but this isn't necessary for the small network described in the question.) Option A describes the procedure to locate a time server for the first computer configured (`gateway.pangaea.edu`) but not for subsequent computers. Although configuring other computers to use `ntp.example.com` instead of or in addition to `gateway.pangaea.edu` is possible, doing so will needlessly increase your network traffic and the load on the `ntp.example.com` server. Thus options B and C are both incorrect. Contrary to option E, NTP is suitable for use on small local networks, and in fact it's very helpful if you use certain protocols, such as Kerberos.

16. B, D. The `cron` utility is a good tool for performing tasks that can be done in an unsupervised manner, such as deleting old temporary files (option B) or checking to see that disk space is not low (option D). Tasks that require interaction or do not occur on a scheduled basis, such as creating accounts (option C), aren't good candidates for `cron` jobs, which must execute unsupervised and on a schedule. Although a `cron` job could restart a crashed server, it's not normally used to start a server when the system boots (option A); that's done through system startup scripts or a super server. Sending files to a printer (option E) is generally handled by a print server such as the `cupsd` daemon.

17. B. User `cron` jobs don't include a username specification (`tbaker` in options A and C). The `*/2` specification for the hour in options C and D causes the job to execute every other hour; the `7,19` specification in options A and B causes it to execute twice a day, on the 7th and 19th hours (in conjunction with the `15` minute specification, that means at 7:15 a.m. and 7:15 p.m.). Thus, option B provides the correct syntax and runs the job twice a day, as the question specifies, whereas options A, C, and D all get something wrong. Option E causes the job to run once an hour, not twice a day.

18. B. The anacron program is a supplement to cron that helps ensure that log rotation, daily backups, and other traditional cron tasks are handled even when the computer is shut down (and, hence, when cron isn't running) for extended periods of time. This is the program to add to the system to achieve the stated goal, and option B is correct. There is no common Linux utility called tempus, so option A is incorrect. Option C's crontab is the name of a file or program for controlling cron, which is likely to be an unreliable means of log rotation on a laptop computer. The ntpd program (option D) is the NTP daemon, which helps keep the system clock in sync with an external source. Although running ntpd on a laptop computer is possible, it won't directly help with the task of scheduling log rotation. The syslog-ng package is an alternative system log daemon, but this program doesn't help solve the problem of missed daily backups when using standard cron utilities, so option E is incorrect.
19. E. The at command runs a specified program at the stated time in the future. This time may be specified in several ways, one of which is teatime, which stands for 4:00 p.m. Thus, option E is correct. The objections stated in options A, B, C, and D are all invalid. (You *may* pass a script to at with the -f parameter, but this isn't required, contrary to option D.)
20. A, C. The contents of /etc/cron.daily are automatically run on a daily basis in most Linux distributions, and the crontab utility can create user cron jobs that run programs at arbitrary time intervals, so both A and C are correct. The at command noted in option B can be used to run a program a single time, but not on a regular basis (such as daily). Option D's run-parts utility is used by some distributions as a tool to help run programs in the /etc/cron.* subdirectories, but it's not used to schedule jobs. Although the crontab program can maintain user crontabs, it's not used as shown in option E and it has no -d parameter at all.

Chapter 8: Configuring Basic Networking

1. A, B, E. Ethernet (option B) is currently the most common type of wired network hardware for local networks. Linux supports it very well, and Linux also includes support for Token Ring (option A) and Fibre Channel (option E) network hardware. DHCP (option C) is a protocol used to obtain a TCP/IP configuration over a TCP/IP network. It's not a type of network hardware, but it can be used over hardware that supports TCP/IP. NetBEUI (option D) is a network stack that can be used instead of or in addition to TCP/IP over various types of network hardware. Linux doesn't support NetBEUI directly.
2. B. IP addresses consist of four 1-byte numbers (0–255). They're normally expressed in base 10 and separated by periods. 63.63.63.63 meets these criteria, so option B is

correct. 202.9.257.33 includes one value (257) that's not a 1-byte number, so option A is incorrect. 107.29.5.3.2 includes five 1-byte numbers, so option C is incorrect. 98.7.104.0/24 (option D) is a network address—the trailing /24 indicates that the final byte is a machine identifier, and the first 3 bytes specify the network. Option E, 255.255.255.255, meets the basic form of an IP address, but it's a special case—this is a broadcast address that refers to *all* computers rather than to the *single* computer specified by the question.

3. C. The gateway computer is a router that transfers data between two or more network segments. As such, if a computer isn't configured to use a gateway, it won't be able to communicate beyond its local network segment, making option C correct. A gateway is not necessary for communicating with other systems on the local network segment, so option A is incorrect. If your DNS server is on a different network segment, name resolution via DNS won't work, as stated in option B; however, other types of name resolution, such as `/etc/hosts` file entries, will still work, and the DNS server might be on the local network segment, so option B is incorrect. Gateways perform the same function in both IPv4 and IPv6 networking, so option D is incorrect. DHCP functions fine without a gateway, provided that a DHCP server is on the same local network segment as its clients (as is normally the case), so option E is incorrect.
4. D. The Secure Shell (SSH) protocol uses port 22, so if the traffic to port 22 is using the correct protocol, it's SSH traffic and option D is correct. The Hypertext Transfer Protocol (HTTP; option A) is conventionally bound to port 80; the Simple Mail Transfer Protocol (SMTP; option B) uses port 25; Telnet (option C) uses port 23; and the Network News Transfer Protocol (NNTP; option E) uses port 119. None of these would normally be directed to port 22.
5. D. The Interactive Mail Access Protocol (IMAP) is assigned to TCP port 143. Ports 21, 25, 110, and 443 are assigned to the File Transfer Protocol (FTP), the Simple Mail Transfer Protocol (SMTP), the Post Office Protocol version 3 (POP3), and the Hypertext Transfer Protocol over SSL (HTTPS), respectively. Although some IMAP server programs also support POP3 and might therefore listen to both ports 110 and 143, the question specifies IMAP exchanges, so option D is the only correct answer.
6. C, E. Option C, `dhcpd`, is the Linux DHCP *server*. Option E, `ifconfig`, can be used for network configuration but is not itself a DHCP client. The others are all DHCP clients. Any given computer will use just one DHCP client (or none at all), but any one of A, B, or D will be available choices.
7. B, C. When used to display information on an interface, `ifconfig` shows the hardware and IP addresses (options B and C) of the interface, the protocols (such as TCP/IP) bound to the interface, and statistics on transmitted and received packets. This command does *not* return information about programs using the interface (option A), the hostname associated with the interface (option D), or the kernel driver used by the interface (option E).
8. A. The `host` program (option A) is a commonly used program to perform a DNS lookup. There is no standard `dnslookup` program (option B), although the `nslookup`

program is a deprecated program for performing DNS lookups. pump (option C) is a DHCP client. ifconfig (option D) is used for configuration of networking parameters and cards. netstat (option E) is a general-purpose network diagnostic tool.

9. B. To add a default gateway of 192.168.0.1, the command would be **route add default gw 192.168.0.1**, as in option B. Specifying the IP address of the host system (as in options A, C, and D) is not necessary and in fact will confuse the route command. Although route provides a -host option, using host (without a dash), as in option E, is incorrect. Furthermore, option E omits the critical add parameter.

10. A, B. The dhclient utility, if installed, attempts to configure and bring up the network(s) passed to it as options (or all networks if it's given no options) using a DHCP server for guidance. Thus option A may work, although it won't work if no DHCP server is available. Option B applies whatever network options are configured using distribution-specific tools and brings up the network. Thus options A and B both may work, although neither is guaranteed to work. Option C displays the network status of eth1, but it won't activate eth1 if it's not already active. There is no standard network utility in Linux, so option D won't work. The netstat utility is a network diagnostic tool; it won't bring up a network interface, so option E is incorrect.

11. E. Although not all systems use /etc/hostname, option E correctly describes it for those systems that use it. The file or files that hold information on package repository servers vary from one package system to another, so option A is incorrect. Option B describes the purpose of /etc/resolv.conf. Option C describes the purpose of /etc/hosts. Option D doesn't describe any standard Linux configuration file, although the gateway computer's IP address is likely to appear in a distribution-specific configuration file.

12. C. The traceroute command (option C) identifies the computers that lie between your own computer and a destination computer, along with some very basic information about network packet travel time and reliability. Thus, traceroute can help you track down the source of the described problem—perhaps a router that's critical to reaching all of the non-responsive systems has failed. The netstat and ifconfig utilities of options A and D both provide information about local network configuration options, but they most likely won't be of much help in diagnosing a problem that affects only some sites. The ping utility (option B) may help you quickly identify sites that have failed but won't be of much use beyond that. You can use dig (option E) to obtain information on the mapping of hostnames to IP addresses, but it won't help in resolving basic connectivity problems.

13. B. Both global and link-local IPv6 addresses can use the system MAC address as part of the IPv6 address, thus option A is incorrect. The fee network address identifies a site-local address but not a link-local address, so option C is also incorrect. An address that starts with 2001 would be a normal global address, making option D incorrect. IPv6 link-local addresses start with fe80, thus C is the correct answer.

14. C. The netstat program produces various network statistics, including the process IDs (PIDs) and names of programs currently accessing the network when it's passed the -p

parameter, thus option C is correct. The `ifconfig` program can't produce this information, and the `-p` option to this program is fictitious, so option A is incorrect. Option B's `/proc/network/programs` file is also fictitious. Option D's `/etc/xinetd.conf` file is real and may provide some information about some servers that are using the network (as described in Chapter 10), but this file won't provide information about all servers, much less about clients that are accessing the network. The `dmesg` command displays the kernel ring buffer, which doesn't contain information on programs that are currently accessing the network, so option E is incorrect.

15. A, D. If you get any response at all, you know that the basic network connection is working, including that the server is responding to the client. With basic knowledge of IMAP commands, `telnet` enables you to test the server's responses in more detail than most IMAP clients (mail readers) permit. Thus options A and D are both correct. Option B describes the functionality of `traceroute` or `tracepath`; `telnet` provides no information about intermediate routers' functionality, so option B is incorrect. Because neither `telnet` nor IMAP on port 143 uses encryption, option C is incorrect. Furthermore, a packet sniffer is likely to have no effect on the transfer of data; it just copies the data so that the packet sniffer's user can see it. Although `telnet` can be used for remote access in a way that could make option E correct, the question specifies using `telnet` to connect to port 143, which is the IMAP port, not the Telnet port. Thus, option E is incorrect. (Furthermore, using `telnet` for remote administration is very risky because `telnet` is an unencrypted protocol.)

16. B. The computer's IP address (172.25.78.89) and netmask (255.255.255.0) mean that the computer can directly address computers with IP addresses in the range of 172.25.78.1 to 172.25.78.254, but the gateway address (172.25.79.1) is outside of this range. Thus, either the IP address or the gateway address is wrong, and option B is correct. Nothing about the way DNS operates necessitates that the DNS server be on the same network segment as the DNS client, so option A is incorrect. Although private IP addresses are often isolated from the Internet, as option C specifies, Network Address Translation (NAT) can get around this limitation. Thus, although there could be some truth to option C, it's not certain to be true. The Class A/B/C distinctions are just guidelines that can be overridden by specific configurations. Thus option D is incorrect. Option E's assertion that `ifup` is used only on computers that use DHCP is incorrect; `ifup` can work on computers that use static IP addresses provided the relevant information is entered correctly.

17. E. The `-n` option is used when you want to use `route` to display the current routing table, and it does as option E specifies. There is no `route` parameter that behaves as options A or C specify. Option B describes the purpose of the `netmask` parameter to `route`. Option D describes the purpose of the `-net` parameter to `route`.

18. E. Option E correctly identifies the function of `/etc/resolv.conf`. Option A describes the purpose of `/etc/services`. Various distribution-specific configuration files perform the function described in option B, but `/etc/resolv.conf` is not one of these files. A DHCP client sends a broadcast to locate a DHCP server; there is no client

configuration file that holds the DHCP server's address, as option C describes. The routing table is maintained internally, although basic routing information may be stored in distribution-specific configuration files, so option D is also incorrect.

19. B. The `/etc/hosts` file holds mappings of IP addresses to hostnames, on a one-line-per-mapping basis. Thus option B is correct. The file does not list the users (option C) or other hosts (option A) allowed to access this one remotely, affect remote administration through a web browser (option D), or map port numbers to protocols (option E).

20. D. The `/etc/nsswitch.conf` file controls the order of name resolution, among other things. Option D correctly describes the procedure for changing the order in which Linux performs name resolution. The `/etc/resolv.conf` file mentioned in option A controls the DNS servers that Linux consults, but it doesn't control access to `/etc/hosts`. Option B's `nslookup` command resolves a hostname, so option B will return the IP address of the computer called `dns`, if Linux can find such a system. The `/etc/named.conf` file of option C is the configuration file for the standard name server. This server isn't likely to be installed on most Linux systems, and even if it is, the procedure described in option C is invalid. Like option B's `nslookup`, option E's `dig` looks up hostname-to-IP-address mappings, so option E will display such mappings for the computers called `local` and `dns`, if they exist.

Chapter 9: Writing Scripts, Configuring Email, and Using Databases

1. D. The `PS1` environment variable contains various formatting codes preceded by a backslash (`\`) as well as text to be included in the primary command prompt. Therefore, option D is correct. There is no environment variable called `PROMPT`, nor is there an environment variable called `PSI`, so options A and B are incorrect. Programs that use a pager, such as `less` or `more`, use the `PAGER` environment variable. If the variable is set, the programs use the pager listed in the variable. Therefore, option C is incorrect. Option D is correct, so option E is incorrect.

2. A. The `alias` built-in command creates a duplicate name for a (potentially much longer) command. Option A shows the correct syntax for using this built-in command. It causes the new alias `cdpt` to work like the much longer `cd ~/papers/trade`. The `export` command in option B creates an environment variable called `cdpt` that holds the value `cd ~/papers/trade`. This will have no useful effect. Option C, if placed in a `bash` startup script, will cause an error because it uses incorrect `alias` command syntax, as does option D. Although `env` is a valid command, it's used incorrectly in option E, and so this option is incorrect.

3. E. Some programs use the `EDITOR` environment variable as described in option E. Contrary to option A, the `EDITOR` environment variable has nothing to do with command-line editing. When you're typing at a bash command prompt, bash itself provides simple editing features, so option B is incorrect. (You can launch the editor specified by $EDITOR by typing Ctrl+X followed by Ctrl+E, though.) The edit command doesn't behave as option C suggests. (This command may be configured differently on different systems.) You can create links called GUI and TEXT to have the EDITOR environment variable behave as option D suggests, but this isn't a normal configuration.

4. C. The PWD environment variable holds the present working directory, so option C is correct. The PATH environment variable (option A) holds a colon-delimited list of directories in which executable programs are stored so that they may be run without specifying their complete pathnames. There are no standard CWD, PRESENT, or WORKING environment variables, so options B, D, and E are all incorrect.

5. A, C. Option A creates the desired environment variable. Option C also creates the desired environment variable. It combines the variable setting and the export of the MYVAR variable using a different method than option A uses. It combines the two commands on one line using a semicolon (;). Option B creates a local variable—but not an environment variable—called MYVAR, holding the value mystuff. After typing option B, you can also type **export MYVAR** to achieve the desired goal, but option B by itself is insufficient. Option D displays the contents of the MYVAR variable and also echoes mystuff to the screen, but it doesn't change the contents of any environment variable. Option E's setenv isn't a valid bash command, but it will set an environment variable in tcsh.

6. E. The ~/.bashrc file is a non-login bash startup script file. As such, it can be used to alter a user's bash environment, and option E is correct. The /etc/inputrc file is a global bash configuration file for keyboard customization and setting terminal behavior. The ~/.inputrc file is for users to create or modify their own keyboard configuration file. Therefore, option A is incorrect. The /etc/bashrc file is a global bash startup script. Editing it will modify users' bash environments, but an individual user should not be able to modify it, so option B is incorrect. There is no standard $HOME/bashrc file because the filename is missing its prefixed period (.). Thus, option C is incorrect. Likewise, option D's $HOME/.profile_bash doesn't refer to a user's configuration file and is incorrect. However, there is a $HOME/.bash_profile bash configuration file.

7. A, D. The env command displays all defined environment variables, so option A satisfies the question. (In practice, you might pipe the results through grep to find the value of a specific environment variable.) The echo command, when passed the name of a specific environment variable, displays its current value, so option D is also correct. DISPLAY is an environment variable, but it's not a command for displaying environment variables, so option B is incorrect. You can use the export command to create an environment variable but not to display the current settings for one, so option C is incorrect. Option E's cat command concatenates files or displays the contents of a file to the screen, but it doesn't display environment variables.

8. B. Before using the ./ execution method, the script must have at least one executable bit set. Therefore, an error will be generated since chmod was not used to modify the execute permissions on the a_script file. Thus Option B is the correct choice since it

would *not* work. Option A uses the `bash` command to execute a script, and this will work fine without any file permission changes. Likewise, when you source a file using either the `source` command or a dot (`.`) and a space, there is no need to modify a scripts permission bits before executing the file. Therefore, option C and option D are incorrect because they also work fine.

9. C. The `cp` command is the only one called in the script, and that command copies files. Because the script passes the arguments (`$1` and `$2`) to `cp` in reverse order, their effect is reversed—where `cp` copies its first argument to the second name, the `cp1.sh` script copies the second argument to the first name. Thus, option C is correct. Because the order of arguments to `cp` is reversed, option A is incorrect. The `cp` command has nothing to do with compiling (option B) or converting (option D) C or C++ programs, so neither does the script. The reference to `/bin/bash` in the first line of the script identifies the script itself as being a `bash` script; it does not cause the arguments to the script to be run as `bash` scripts, so option E is incorrect.

10. E. The commands iterated by the `for`, `while`, and `until` loops are located between the `do` and `done` constructs. Therefore, option E is correct. Commands in the `then` statement section are for an `if-then` construct, not a loop, thus option A is incorrect. Double semicolons are used for `case` constructs, but not loops, and so option B is incorrect. The `case` and `esac` keywords begin and end a `case` construct, and thus option C is incorrect. A test statement can be used to determine whether or not a loop's commands should iterate or not. However, it does not contain the actual commands to be iterated, and therefore option D is incorrect.

11. B, C. Valid shell scripts begin with the characters `#!` and the complete path to a program that can run the script. Options B and C both meet this description, because `/bin/bash` is a shell program that's installed on virtually all Linux systems and `/bin/tcsh` is often also available. There is no standard `/bin/script` program, so option A is incorrect. Options D and E are both almost correct; `/bin/sh` is typically linked to a valid shell and `/bin/zsh` is a valid shell on many systems, but the order of the first two characters is reversed, so these options are incorrect.

12. A, B, D. The `for`, `while`, and `until` statements are all valid looping statements in `bash`, so options A, B, and D are all correct. The `if-then` statement in `bash`'s scripting language tests a condition and, if it is true, executes its commands *one* time only. Therefore, option C is incorrect. The `case` statement is a conditional, not a looping statement in `bash`, so option E is incorrect.

13. B. When aliases are properly configured, any email addresses sent to the email with an alias is received by the alias account. Therefore, option B is correct. The `postmaster` username would not receive the email because the alias is set to `john`, and so option A is incorrect. The `~/.forward` file is associated with email forwarding, not aliases. Therefore, option C is incorrect. There is no reason for `root` to receive this email, so option D is incorrect. An alias *does* allow email to be sent to the alias account, so the statement in Option E does not make sense and is incorrect.

14. C. The Fetchmail program is a tool for retrieving email from remote POP or IMAP servers and injecting it into a local (or remote) SMTP email queue. As such, it's not an SMTP server, so option C is correct. Postfix (option A), sendmail (option B), Exim (option D), and qmail (option E) are all popular SMTP email servers for Linux.

15. B. The `-s` option to `mail` sets the message subject line, and `-c` sets carbon copy (`cc:`) recipients. Input redirection (via `<`) reads the contents of a line into `mail` as a message. A `mail` command line normally terminates with the primary recipient. Thus, option B correctly describes the effect of the specified line. Options A, C, D, and E are all confused in their interpretation of the effects of `mail` parameters. Options A, B, and D also confuse input and output redirection, and option A incorrectly suggests that a script (or the `mail` program) can elevate its run status to `root` privileges.
16. D. To view your mail queue, use the `mailq` command (option D). The `service sendmail status` command is a SysV service status command and does not show mail queues, so option A is incorrect. Option B is a printer command and is therefore incorrect. Option C is close, but the correct command is `sendmail -bp` not `-bq`. Option E will show you the various directories within `/var/spool` and is therefore not the correct command.
17. B. The `/etc/aliases` file configures system-wide email forwarding. The specified line does as option B describes. A configuration like this one is common. Option A has things reversed. Option C is not a valid conclusion from this evidence alone, although an intruder conceivably may be interested in redirecting `root`'s email, so if `jody` shouldn't be receiving `root`'s email, this should be investigated further. Although the effect of option D (`jody` reading `root`'s email) is nearly identical to the correct answer's effect, they are different; `jody` cannot directly access the file or directory that is `root`'s email queue. Instead, the described configuration redirects `root`'s email into `jody`'s email queue. Thus, option D is incorrect. Because `/etc/aliases` is an email configuration file, not an account configuration file, it can't have the effect described in option E.
18. B. The `CREATE DATABASE` command creates a new database with the specified name. Because SQL commands are case insensitive, this command may be typed in uppercase or lowercase, and option B is correct. Options A and C both use the incorrect command `NEW` rather than `CREATE`, and option C specifies the database name as `FISH` rather than `fish`. (Database names *are* case sensitive.) Option D reverses the order of the `CREATE` and `DATABASE` keywords. Option E uses the fictitious command `DB`.
19. A, D. A single database may hold multiple tables, as option A suggests. Option D is also correct; if data is split across tables (such as into tables describing objects generically and specifically), databases can be more space efficient. Option B is incorrect because the `DROP` command doesn't combine tables—it *deletes* a table! Option C is incorrect because it reverses the meaning of rows and columns in a SQL table. A lossy compression algorithm, as the name suggests, deliberately corrupts or loses some data—an unacceptable option for a text database, making option E incorrect. (Lossy compression is used for some audio and video file formats, though.)
20. C. The `UPDATE` command modifies existing database table entries, and in this case it does so as option C describes. Option B also describes an update operation, but in a

confused and incorrect way. Options A and D both describe database retrieval operations, but `UPDATE` doesn't retrieve data. Option E mistakenly identifies `stars` as a database name, but it's a table name, and it mistakenly identifies the operation as adding a new entry (`INSERT` in SQL) rather than as modifying an existing entry (`UPDATE` in SQL).

Chapter 10: Securing Your System

1. E. The server names alone are insufficient to determine whether they're legitimate. The computer in question may or may not need to run any of these servers, and their presence may or may not be intentional, accidental, or the sign of an intrusion. Thus, option E is correct. Contrary to option A, the mere presence of an SSH server does not ensure security. Although, as option B asserts, FTP is not a secure protocol, it's still useful in some situations, so the mere presence of an FTP server is not, by itself, grounds for suspicion. Similarly, in option C, although some administrators prefer Postfix or qmail to `sendmail` for security reasons, `sendmail` isn't necessarily bad, and the names alone don't guarantee that the `sshd` and `proftpd` servers are legitimate. As option D states, `sendmail` and `proftpd` both use unencrypted text-mode transfers, but this is appropriate in some situations, so option D is incorrect.
2. C. Although Nmap and other port scanners are useful security tools, troublemakers also use them, and many organizations have policies restricting their use. Thus, you should always obtain permission to use such tools prior to using them, as option C specifies. A port scanner can't cause damage to `/etc/passwd`, so there's no need to back it up, contrary to option A. A port scanner also doesn't need the `root` password on a target system to operate, so you don't need this information, making option B incorrect. (In fact, asking for the `root` password could be seen as extremely suspicious!) Although you could use `sudo` to run Nmap, there's no need to do so to perform a TCP scan, and you can perform a UDP scan by running Nmap as `root` in other ways (such as via a direct login or by using `su`). Thus, option D isn't strictly necessary, although you might want to tweak `/etc/sudoers` as a matter of system policy. Because a firewall is part of your network's security, you probably want it running when you perform a network scan, contrary to option E. Furthermore, it would be safer to leave the firewall running and scan from behind it if you want to test the security of the network in case of a firewall breach.
3. C. The `/etc/security/limits.conf` (option C) file holds the configuration settings that allow you to limit users' access. The other options listed don't give the correct path to this file.
4. A, B, C. Nmap (option A) is usually used to perform scans of remote computers, but it can scan the computer on which it's run as well. The `netstat` (option B) and `lsof` (option C) utilities can both identify programs that are listening for connections (that is, open ports) on the local computer. The Network File System (NFS) and some other

servers use the `portmap` program (option D), but it's not used to identify open ports. There is no standard Linux `services` program (option E), although the `/etc/services` file holds a mapping of port numbers to common service names.

5. B. The `-perm` option to `find` locates files with the specified permissions, and `+4000` is a permission code that matches SUID files. The `-type f` option restricts matches to files in order to avoid false alarms on directories. Option B uses these features correctly. Options A, C, and D use these features incorrectly. Option E specifies a fictitious `-suid` parameter to `find`.
6. A. Option A correctly describes the meaning of the specified line. A percent sign (`%`) identifies a Linux group name, and the remainder of the line tells `sudoers` to enable users of that group to run all programs as `root` by using `sudo`. The remaining options all misinterpret one or more elements of this configuration file entry.
7. B. The `netstat` command can do what is described in the question. The `-ap` options to the command are good choices to discover all the open network connections, so option B is correct. Although `lsof` can also accomplish the job, the `-c a` option is incorrect; this option restricts output to processes whose names begin with `a`. Thus, option A is incorrect. Option C's `ifconfig` command doesn't display open network connections, so it's incorrect. Although option D's `nmap` command will locate ports that are open on the `localhost` interface, it doesn't locate all open *connections*, nor does it locate connections on anything but the `localhost` interface. Option E's `top` command displays a list of processes sorted by CPU use, not open network connections (`-net` is an invalid option to `top` as well).
8. D. Option D is correct. TCP wrappers uses this feature to allow you to override broad denials by adding more specific access permissions to `hosts.allow`, as when setting a default deny policy (`ALL : ALL`) in `hosts.deny`.
9. C. The `bind` option of `xinetd` lets you tie a server to just one network interface rather than link to them all, so option C is correct. It has nothing to do with running multiple servers on one port (option A), specifying computers by hostname (option B), resolving conflicts between servers (option D), or the Berkeley Internet Name Domain (BIND) or any other DNS server (option E).
10. A, D. Using a firewall rule to block Waiter's port, as in option A, can increase security by providing redundancy; if Waiter is accidentally run in the future, the firewall rule will block access to its port. Uninstalling the program, as in option D, improves security by reducing the risk that the program will be accidentally run in the future. Most programs don't have a "stealth" mode, so option B is incorrect. (Furthermore, *reading* the documentation isn't enough; to improve security, you must change some configuration.) Tunneling Waiter's connections might have some benefit in some situations, but this configuration requires setup on both client and server computers and by itself leaves the server's port open, so option C is incorrect. Clients associated with the server program, installed on the server computer, pose little or no risk of abuse of the associated server; the clients on *other* computers are most likely to be used to abuse a server program, and you can't control that. Thus option E is incorrect.

11. B. Option B correctly describes how to accomplish this goal. Option A is incorrect because the `hosts_allowed` option isn't a legal `xinetd` configuration file option. Option C correctly describes how to configure the described restriction using TCP wrappers, which is generally used with `inetd`, but it's not the way this is done using `xinetd`. Option D is also a TCP wrappers description, but it reverses the meaning. Option E's `iptables` utility configures a firewall. Although a firewall rule could be a useful redundant measure, the question specifies an `xinetd` configuration, and option E's use of `iptables` is incorrect.
12. B. Ideally, passwords should be completely random but still memorable. Option B's password was generated from a personally meaningful acronym and then modified to change the case of some letters, add random numbers and symbols, and extend its length using a repeated character. This creates a password that's close to random but still memorable. Option A uses a well-known mythological figure, who is likely to be in a dictionary. Option C uses two common words, which is arguably better than option A, but not by much. Option D uses two closely related words separated by a single number, which is also a poor choice for a password. Option E uses a sequential series of numbers, which is a poor (but sadly common) password choice.
13. A. Phishing (option A) involves sending bogus email or setting up fake websites that lure unsuspecting individuals into divulging sensitive financial information or other sensitive information. Script kiddies (option B) are intruders who use root kits. Spoofing (option C) involves pretending that data is coming from one computer when it's coming from another. Ensnaring (option D) isn't a type of attack. Hacking (option E) refers to either lawful use of a computer for programming or other advanced tasks or breaking into computers.
14. C. The `/etc/nologin` file, if present, prevents logins from ordinary users; only `root` may log in. You might set this file when performing maintenance and then forget to remove it, thus explaining the symptoms in the question. Thus, option C is correct. The `syslogd` daemon mentioned in option A records system messages, and it is unlikely to produce the specified symptoms. The login process ordinarily runs as `root` and is normally SUID `root`, so options B and D are also incorrect. Shadow passwords, as in option E, are used on almost all modern Linux systems and are not likely to cause these symptoms.
15. B, C. SSH is most directly a replacement for Telnet (option B), but SSH also includes file-transfer features that enable it to replace FTP (option C) in many situations. SSH is not a direct replacement for the Simple Mail Transfer Protocol (SMTP, option A), the Network Time Protocol (NTP, option D), or Samba (option E).
16. A . The `ssh_host_dsa_key` file holds one of three critical private keys for SSH. The fact that this key is readable (and writeable!) to the entire world is disturbing, so option A is correct. In principle, a troublemaker who has acquired this file might be able to redirect traffic and masquerade as your system, duping users into delivering passwords and other sensitive data. Because of this, option B (no) is an incorrect response, and

the conditions imposed by options C, D, and E are all irrelevant, making all of these options incorrect.

17. B. SSH protocol level 2 is more secure than protocol level 1; thus option B (specifying acceptance of level 2 only) is the safest approach. Option A is the *least* safe approach because it precludes the use of the safer level 2. Options C and D are exactly equivalent in practice; both support both protocol levels. Option E is invalid.

18. E. Allowing only normal users to log in via SSH effectively requires two passwords for any remote `root` maintenance, improving security, so option E is correct. Whether or not you permit `root` logins, the SSH server must normally run as `root`, since SSH uses port 22, a privileged port. Thus, option A is incorrect. SSH encrypts all connections, so it's unlikely that the password, or commands issued during an SSH session, will be intercepted, so option B isn't a major concern. (Nonetheless, some administrators prefer not to take even this small risk.) SSH doesn't store passwords in a file, so option C is incorrect. Because SSH employs encryption, option D is incorrect (this option better describes Telnet than SSH).

19. D. Option D provides the correct command to import `fredkey.pub` prior to use. The `inspect-gpg`, `import-gpg`, and `gpg-import` commands of options A, C, and E are fictitious, and there is no `--readkey` option to `gpg`, as option B suggests.

20. E. The usual method of sending encrypted messages with GPG entails the sender using the recipient's public key to encrypt the message. Thus, option E is correct. Option A would be correct if your correspondent needed to send you an encrypted message, but the question only specifies you sending the encrypted message. Options B, C, and D all entail delivery of private keys, which is inadvisable at best, because private keys in the wrong hands permit the holder to impersonate the person who owns the keys.