SERVER IN A BOX

MANUAL

For EMAC Linux distribution v2.4

MANUAL REV. 2.4.2

Introduction

The EMAC Server-In-a-Box (SIB) is designed to act as a general-purpose embedded Linux server. The SIB runs the Linux operating system kernel (a UNIX work-alike), and as such retains all of the networking and communications capabilities one would expect of such an operating system. The SIB v2.0 includes Linux kernel 2.4.9 and GNU standard C libraries version 2.2.3 (glibc6 v2.2.3).

Right out of the box, the SIB is configured for ethernet, serial IP, serial terminal, and raw serial connections. It includes a web server, an FTP server, a TELNET server, and a periodic command scheduler. Other devices, including PPP modem links, may be added and configured by the user. Furthermore, the SIB can be set up as an NFS server, an SMB (MS Windows file sharing) server, a mail relay, a firewall, and many other configurations.

This manual is designed to help you configure and work with the Linux system, and understand what you do and why. The first section of this manual, Features, explains what the SIB can do, how it does it, and how each family of utilities can be set up and configured. The second section of this manual, Configuration, explains the details of setting up the SIB using the EMAC configuration utility. It is hoped that this format will first help the user understand what to configure and why, before explaining exactly how. Users with thorough knowledge of Linux should at least skim section one briefly before moving on to section two, as section one contains important information about differences between the EMAC SIB and other (full-sized) Linux distributions.

Users and Security

Linux follows the UNIX security model of separate users and username / password authentication. This means that each person to use the machine has a separate "account," distinguished by a username (also called a "login name"). To gain access to an account on a UNIX or Linux machine, a person must know both a username and the password associated with it.

The UNIX security model also includes the concept of groups. Each account (user) is a member of one or more groups, and each group has one or more members. Groups allow multiple people to access the same file(s), yet still prevent non-privileged users from viewing or modifying those files.

Each user (and each group) can have different permissions and privileges. These privileges take the form of "file permissions." Each file in a UNIX system has two ownerships. Every file is owned by a single user, and by a group. Each file also has three sets of three permissions: read, write, and execute permissions may be specified separately on each file for the file's user owner, the file's group owner, and everyone on the entire system. These permissions are expressed by the following bitfield:

```
User                Group               Everyone
read-write-execute-read-write-execute-read-write-execute
```

When describing permissions, this bitfield is expressed as an octal (base 8) number. For example, permissions 754 mean that the user owner of the file may read, write, or execute the file (7). The group may read or execute, but not write (5). Other users on the system may read the file, but not write it or execute it (4).

Nearly every important file on a UNIX system is owned by a single user and group, root (root user, root group). The root account is essential to the security of any UNIX system. Having all of the important executable and configurations files on the system owned by root means that only the root user may alter the way the system works. Other users have no permission to execute or change system files, and therefore they cannot modify or damage the system. As such, it is critical that the root password be kept secret, and changed as often as is practical. A clumsy or malicious individual with the SIB's root password could

render the entire system inoperative in a matter of seconds. Every SIB ships from the factory with the same root password–this means that anyone who has ever purchased an SIB knows your root password! EMAC recommends you change the SIB's root password as soon as you've finished reading the manual.

Aside from the root account, which is present on every system, the SIB has one additional account configured. The username is "`www`" (no quotes). This user does not have privileges to change the system configuration. Instead, this account is designed to allow a semi-privileged user to upload HTML files to the SIB. From there they may be served by the SIB's web server. This account may also be used when transferring new executables to the SIB. While it is much more difficult for an individual with the www password to damage the SIB, it is still possible. EMAC therefore recommends you change www's password right after you change the root password.

Command-line Tools

Certain command-line utilities are present on nearly every UNIX system. The SIB provides as many of these as is practical in the space available. The functionality of these basic commands is provided by a pair of binaries, `/bin/busybox` and `/bin/tinylogin`. However, it is important to note that some of these condensed command line tools function somewhat differently than the versions you might find in a full-sized Linux distribution. If you are already familiar with tools such as `grep` and `sed`, you may find that the syntax differs slightly from what you are used to, or that certain switches are not supported. For further reference, please see the `busybox.html` and `tinylogin.html` files on the SIB. These can be accessed via a web browser, by entering `x.x.x.x/busybox.html` or `x.x.x.x/tinylogin.html`, where x.x.x.x is the IP address of the SIB.

The SIB's main shell is GNU `Bash`, with full scripting capability. Most of the standard command-line tools are provided by `busybox`, a multi-call binary which does the work of many different programs, including `grep`, `sed`, `awk`, and `ash`. In some cases, we felt that the Busybox implementation of a particular function was inadequate (`tar`, for example). In these cases, regular versions of these programs from full-sized distributions are used. Descriptions of some of the most common command line utilities follow. Keep in mind when running these programs that Linux (and UNIX in general) is case-sensitive; capitalization matters. In all cases, the program name must be types in all lowercase letters.

The `ls` utility works much like the `dir` command in DOS. With no parameters, `ls` will list the files in the current directory. A directory name may also be specified on the command line, `ls` will then list this directory instead. For more detail, the "`-l`" switch (dash-ell, no quotes) may be specified between `ls` and the directory name. This will display information about files including ownership, permissions, size, and modification times.

The `cd` command changes the current directory, exactly as in DOS. As UNIX has no concept of "drives," all files are kept in a single unified file system. Forward slashes (`/`) are used to separate subdirectory names.

Each user has a "home directory." This directory is where a user will begin after logging in. For the root user, this directory is `/root/`. For other (regular) users, this directory will usually be `/home/<username>/`. When using the command line, a user's home directory may be abbreviated by a tilde character ('~'). For example, "`cd ~`" will bring you to your home directory. The command "`cd ~foo`" will take you to user foo's home directory (assuming you have permission to read that directory). Similarly, "`~foo/bar`" will execute a program called bar in foo's home directory. Note that a tilde abbreviation should never be prefixed with a '`/`'.

The chown, chgrp, and chmod commands modify file ownerships and permissions. The chown utility will change the owner of a file to one specified on the command line. Chgrp will change the group owner of a file in the same manner. The chmod utility changes the permissions of a file (chmod <permissions> <file>). The new permissions are specified numerically, as described previously. Note that you must have write permission on a file to use these utilities on it.

The cat utility will display the entire contents of a file to the screen. This should only be done to text files, as "catting" binary files can cause the shell to enter odd states in which it prints unreadable characters. There is no way to pause the output from cat. The more utility fixes this shortcoming by displaying a specified file one page at a time. Simply press a key to view the next page of text. The more utility is the standard way to view text files on the SIB.

The SIB includes a simple text editor, called nano. It is invoked from the command line as "nano <filename>". When nano starts, its help screen is displayed on top of the edit window. If for some reason the help screen does not appear, hold down the CONTROL key and press x, then h.

To move a file in the SIB filesystem, use the mv command. Its syntax is identical to the DOS move command. The cp command is likewise similar to the DOS copy utility. Linux does not have a rename function, but mv can be used to change a file's name. The rm command will delete a file. To delete a directory, the "-r" switch must be specified before the directory name to delete. To prevent rm from prompting for confirmation, use the "-f" switch. CAUTION: there is no undelete utility. File deleted from the SIB are gone, permanently.

Loadable Modules and Device Drivers

There two ways to use a device driver in Linux: it can be compiled into the kernel, or it can be compiled separately as a "loadable module." For ease of use, the SIB uses both methods. Drivers compiled into the kernel are always active, and require no user or system input to work. Drivers such as TCP/IP, SLIP, PPP, ramdisk, and serial have been compiled into the kernel.

Modular drivers must be explicitly loaded into the kernel, after the machine has booted. They can be loaded (and unloaded) at any time. Most commonly, they are loaded by boot scripts (similar to the DOS autoexec.bat script). Drivers such as parallel port, DOS filesystem, and ethernet have been compiled as modules. Modules are kept in subdirectories of the /lib/modules/2.2.14/ directory, indexed by type. Modular drivers are loaded by the insmod command, and removed by the rmmod command. All modules currently loaded can be listed by the lsmod command. Note that some drivers require parameters on the command line. In particular, ISA peripherals like network and sound hardware usually need their I/O base address specified.

The SIB automatically loads a number of modules at boot time. Any module listed in the file /etc/modules will be loaded at boot time. Command line parameters for these modules may also be specified. Note that for space reasons, the SIB includes very few of the available modules. If you require a driver that is not included by default, contact EMAC to have the driver sent to you.

Another caveat of using modules is that some modules depend on others, and will not load until the dependencies have been satisfied. The SIB cannot automatically resolve these dependencies, so a user must explicitly specify the loading of any module dependencies before loading the desired modules. Some common examples of dependencies are the NE2000 driver (ne.o), that requires the 8390.o module, and the SLIP and PPP drivers (slip.o and ppp.o), that require the slhc.o module.

Flash disks and RAM disks

The SIB runs its operating system from an M-Systems DiskOnChip. This is in contrast to some other small Linux distributions, which load the filesystem into a RAM disk and execute from there. However, the SIB does use RAM disks for some tasks. Flash media such as the DiskOnChip can only support a limited number or write operations (though they can be read an unlimited number of times.) As such, a RAM disk is used for the directory tree which is most often written to: the `/var` directories.

The `/var` directories contain files which are intended to be changed often. Lock files, temporary files, and log files are all kept in `/var`. Keeping the `/var` directories in a RAM disk saves a great deal of wear on the flash. However, it also means that log files are lost any time the SIB loses power. If you wish to keep log files on a permanent basis, you should use the periodic command scheduler CRON (described later in this document) to copy the logs to flash, or to send them to a remote computer via FTP or e-mail.

As with most other operating systems, a Linux system should generally not be powered off without first shutting down the operating system. This allows Linux to flush its buffers and finish any pending writes to disk (flash). If a Linux system is powered down without warning, data may be lost. In the worst case, a power down could occur while a write is taking place, causing corruption of the filesystem. This is particularly dangerous when using flash, as a flash device must be erased before it can be written, greatly increasing the time in which the media is vulnerable to corruption. The SIB can be properly shutdown by using the command line "`shutdown -h now`".

To work around this problem, it is possible to mount the filesystem read-only. If no writes can be performed, no corruption can occur. Mounting read-only is done in the boot scripts. If the file `/etc/READONLY` exists, the SIB will attempt to mount its root filesystem read-only. However, the user will be given the opportunity to override this each time the system boots. During the boot script sequence, a prompt will appear, and the user will be given five seconds to press a key to override the read-only option. If the key is not pressed, the filesystem will be mounted read-only.

It is also possible to change the filesystem from read-only to read-write and back from the command line. The command "`mount -o remount,rw /dev/diskroot /`" will re-mount the filesystem in read-write mode. Using the same command line, but replacing "`rw`" with "`ro`", will remount the flash read-only.

Note that when the root filesystem is mounted read-only, the `/dev` directory is mounted in a second ramdisk. This allows users to log in even when the filesystem is read-only. If you use the command lines above to remount the flash read-write, changes to `/dev` will be lost when the system loses power.

Login Terminals

There are (at least) four major ways to access the SIB. Each is largely similar to the others, but there are a few important differences. The four terminals described here will be the console, serial terminals, dial-in terminals, and TELNET sessions.

The first way to access the SIB is via the console. If the unit ships without video, this console is accessed by connecting a null modem cable to ttyS0, the first serial port of the SIB. From here, you can give the LILO bootloader boot commands, see startup messages, control mounting the root filesystem readonly or read-write, and login to the SIB. The serial terminal settings are: 9600 baud, 8 data bits, 1 stop bit, no parity, no flow control, and vt100 terminal emulation.

If the unit is equipped with video, this console is accessed in the traditional manner, by connecting a monitor and keyboard to the SIB. In this instance, you may also optionally configure the SIB with a serial console by editing /etc/lilo.conf.doc and uncommenting references to the serial console. Then re-run LILO to update your changes. You may also remove the ability to login via a serial port by editing the file `/etc/inittab`.

When a "dumb terminal" is plugged into this serial port, a login prompt appears, and a user can log into the SIB. Alternately, a second PC can be plugged into `ttyS0` using a NULL modem cable, and a terminal program such as minicom or hyperterminal may be used. It is important that when using a second PC, a "NULL modem" or "crossover" cable is used; a standard serial cable will not work.

The third type of terminal available on all SIBs is the TELNET server. For this to work, the SIB's ethernet port must be plugged into a network, and its networking must be configured properly. In particular, you must set the SIB's IP address and routing data to reflect your LAN. If you are using DHCP, you must know what IP address your DHCP server assigns the SIB. You can test the network setup of the SIB by using the "`ping`" program found on most network operating systems. If the SIB responds on the expected IP address, you can then use the TELNET client included with the operating system on another machine to TELNET to the SIB and log in.

The final login method is only available on SIBs with the modem option installed. These SIBs can be set up to answer the phone, then provide a serial terminal on the modem line. This configuration is also kept in the `/etc/inittab` file and controlled through the master configuration utility. With the exception of answering the phone line, this terminal type behaves exactly as a serial terminal.

To increase security, the root user is only allowed to log in via certain "devices." Every terminal type uses a different type of device. The root user must be explicitly allowed to log in on each device. However, since UNIX/Linux allows for multiple users, there are multiple devices of each type.

The console uses devices `tty0` through `tty3` if the SIB is equipped with video. Otherwise, it will use a serial terminal. Serial terminals use devices `ttyS0 - ttyS3` (these correspond to `COM1 - COM4`). Since modems are generally treated as serial devices, dial-in terminals also use a ttySx device, depending on which serial port the modem occupies (usually `ttyS2`). TELNET uses devices `ttyp0` and so on...there are many `ttyp*` devices, as many users may be logged in via TELNET at once.

The root user may only log in on devices explicitly listed in the file `/etc/securetty`. Should changes need to be made, this file must be edited by hand. By default, several TELNET devices are listed in this file. This aids remote configuration and troubleshooting, but it is also often a security risk. If you do not intend to configure or manage the SIB via TELNET, it is recommended that the TELNET devices (`ttyp*`) be deleted from this file. IT IS IMPORTANT THAT THE CONSOLE AND SERIAL DEVICES NOT BE REMOVED FROM THIS FILE. If that happened, then root would be unable to log in, and configuration and management of the SIB would be impossible.

When a user logs in, certain scripts are automatically run. These scripts will set the user's path. prompt, and other environmental variables. The scripts `/etc/profile`, `~/.profile`, and `~/.bashrc` are all executed, in order. These scripts are by default executed by the `Bash` shell. The scripts in your home directory may be changed should you wish to add to the path, set a command alias, or to run a program every time you log in.

This is the case with the root user: every time the root user logs in, the master configuration utility is started by the ~root/.profile script. Note, however, that only the ~/.bashrc script is executed if a shell is not

the user's login shell. Thus, the master configuration utility is not started again when the root user requests a second shell (bash prompt) from the master configuration utility.

Periodic Command Scheduler

The SIB comes with the standard UNIX periodic command scheduler, `CRON` ("Vixie" implementation). This is a program which starts at boot time and runs in the background, executing specified commands at specified times. Programs can be executable binaries or shell scripts. `CRON` is configured by editing its configuration file `/etc/crontab`, which may be done from the main system configuration utility. The file may also be edited directly.

The `crontab` file consists of zero of more entries with the basic format "at this time, execute this command". Since CRON executes on its own (with no user account configuration), `crontab` may also include statements which set and export the PATH shell variable (the default `crontab` file contains such statements, which should not be modified except by an experienced user).

Times may be specified from the minute up to the month. The specific format for a crontab entry is:

```
<minute> <hour> <day of month> <month> <day of week> <user> <command>
```

All times may be specified by number, and the hour field goes by a 24-hour clock. The day of the week may be specified numerically, by the first three letters of its name. When specified numerically, the day of the week may be `0-7`, with both `0` and `7` representing Sunday. Ranges and lists of times are supported. Thus, placing "`5-10`" in the hour field will cause the command to be executed every hour from five AM to ten AM. The equivalent (comma-separated) list would be "`5, 6, 7, 8, 9, 10`". Note that lists and ranges are not supported when using week day names. Divisors are also supported. For example, "`5-10/2`" in the hour column would execute the given command once every other hour between five and ten AM. Finally, "`any`" or "`all`" may be specified by placing a star ("*") in the field. Fields are separated by whitespace (tabs or spaces).

For example, the line

```
42 0 * 1,5  2-4 root /bin/echo "CRON command is executing" > /dev/console
```

Will print the message "CRON command is executing" to the SIB's console at 12:42 AM every Tuesday through Thursday during January and May. The field "user" contains the user that the command will be executed as. In most cases, this will be **root**. (Note that **root** has special privileges; care should be taken that essential system files are not modified by crontab entries.)

Crontab has several pre-defined entries, designed to perform standard system "housekeeping" tasks. These entries are designed to run every file (usually shell scripts) in a particular /etc subdirectory at regular intervals. These subdirectories are `cron.daily/`, `cron.weekly/`, and `cron.monthly/`. Simply put, everything in `/etc/cron.daily/` will be executed once per day (usually late at night so as not to slow down user operations.) Everything in `/etc/cron.weekly/` will be executed once per week, and so on. This can be a good mechanism to limit the lengths of log files, and basic scripts to manage log files are already in the `CRON` directories. User files placed into these subdirectories will be executed along with the log file management scripts, providing a simple alternative to making actual crontab entries.

Developing Your Own Applications

The Server In a Box is a real Linux system, and as such can run any application that can be run on a

Linux workstation. However, the SIB is not a development platform itself. It has no compiler, no standard headers, no help tools, and a rudimentary text editor.

To develop custom applications for the SIB, it is strongly recommended that the user set up a Linux workstation using one of the modern, standard Linux distributions. RedHat 7.x, Debian 2.3, and many others will work well. For maximum compatibility, EMAC also recommends that your development system use the same kernel and standard C libraries as the SIB (the SIB uses kernel 2.4.9 and glibC 2.2.4). In particular, we cannot guarantee compatibility if your application is compiled on a system with a newer version of glibC.

Many languages and compilers are available for Linux, including C/C++, Fortran, BASIC, assembly, and Perl. Not all languages are supported by default. For example, the default SIB configuration does not include interpreters for BASIC, Perl, Python, or any other interpreted language (shell script excepted). Compiled languages such as C, Fortran, and assembly do not require special interpreters to execute. However, a particular program may need certain "shared objects" added to the system before they will execute.

To save disk space, GNU/Linux systems keep code used by multiple applications in "shared object" libraries. The libraries work much like .dll files in MS Windows. The libraries are usually found in the `/lib`, `/usr/lib`, and `/usr/local/lib` directories, and can be identified by their .so.<version> extensions. The `/lib` directory also contains the dynamic linker, `ld-linux.so.2`.

Because flash space is limited, the SIB only contains the most commonly used libraries, such as the c libraries, network name resolution, and the ncurses library. Many programs will need other libraries to run. If your application will not run, use the command line "`ldd <filename>`" to display a list of the libraries your application needs. If any of the libraries are listed as "not found," then you will need to either copy that library to the SIB, or specify static linking when you compile your program.

It is also important to note that many "libraries" are actually dynamic links to files with similar names. Depending on the compiler, an application may search for a library by several different names. If you copy a library to the SIB and your application still does not find it, you may need to create a link to the library with a different name. Shared objects are also cached, along with their locations. If your application continues to insist that a library does not exist even though it does, you may need to use a program called `ldconfig`. Copy this binary to the SIB from your development workstation, and execute it once, then try your application again – odds are that it will now function properly.

If your program works properly on your development workstation but reports "segmentation faults" when run on the SIB, there are several possibilities. The first is that your program was linked with a newer version of the standard C libraries, and the SIB's libraries are not forward-compatible. This behavior was seen when programs linked against glibc 2.1.2 were run using glibc 2.0.7. To reduce this problem, the SIB was created using the newest libraries available at the time. However technology is sure to progress.

The second major possibility is that your program binary was corrupted in transfer. In particular, the DOS-based FTP client included with MS Windows '95 and '98 default to ASCII mode, meaning that the FTP client may modify the file it is transferring. This FTP client must explicitly be set to transfer binary files. Other FTP clients may have similar behavior.

It can be desirable to have an application start at boot time. There are two ways to accomplish this: via setup scripts, or via `init`. `Init` is that "mother of all processes." Its task is to start login terminals on specified devices, and usually keep them running by re-starting them when they exit. `Init` is

configured by the /etc/inittab file. Init is designed to associate tasks with devices. If your application will use a single serial port or a virtual console terminal, then init may be an appropriate method to run your application each time the SIB boots.

If your application is not associated with a single device, then it should be started via boot scripts. These scripts perform much the same tasks as the DOS autoexec.bat script, though on a much grander scale. SIB boot scripts are stored in the /etc/init.d directory, however they are not executed from that location. When init starts, it executes a script called rc.d. This script in turn starts several other scripts. First, scripts in /etc/rcS.d is executed, in alphabetical order. Only scripts that start with the letter 'S' are executed, and it is common to follow this 'S' with a two-digit number, to control order of execution. Note that these scripts in /etc/rcS.d are actually links to the actual scripts in /etc/init.d.

Once the scripts in /etc/rcS.d have been executed, another directory of scripts is executed. Which directory depends on the "run level" of the system. The run level is a concept used by UNIX to make it easy to control the starting and stopping of different services at boot time. The default run level is set in /etc/inittab. The SIB's default run level is 2. When run level 2 is entered, scripts in /etc/rc2.d are executed. Similarly, when run level 5 is entered, the scripts in /etc/rc5.d will be executed.

When the system leaves one runlevel, it first attempts to "kill" services which are no longer desired, by running scripts in the appropriate directory which begin with 'K'. For example, when the system leaves run level 2, all scripts in /etc/rc2.d beginning with 'K' are executed. The start scripts (scripts that begins with 'S') are then executed in the directory for the new run level.

To start you application at boot time, create a shell script which starts your program and place it in the /etc/init.d directory. Be certain to set execute permission on your script. Then, in the directory associated with the desired run level, create a link to your script. Make sure the link name begins with 'S', and use a number which falls after the rest of the scripts ('S99' is usually safe). Your script will then run after all other system configuration has taken place, and will start your application.

Networking:

Networking addressing

The SIB communicates with other computers using the standard suite of internet protocols (IP). Computers using IP talk to one another by means of an IP address. Each machine on the Internet has a unique IP address, assigned by their network administrator. An IP address usually consists of four numbers (0-255), separated by dots. Examples are 192.168.1.50 and 10.0.2.1.

The SIB has several ways in which it can connect to a network. The SIB can use IP over one or more serial ports (SLIP devices,) modems (PPP devices,) or ethernet devices. These devices are referred to as "interfaces". A particular interface is described by a combination of letters, which describe the type of interface, and a number. Interfaces of a particular type are number from 0, in the order that they were activated. For example, the first ethernet interface activated is identified as "eth0". Similarly, the first SLIP device activated is "sl0," and the first PPP device is "ppp0." Note that eth0, sl0, and ppp0 may all exist on the same machine at the same time.

An SIB is not restricted to having a single IP address. In fact, every network interface may have a different IP address. For example, the eth0 interface may have address 10.0.2.41 (the factory

default). IP packets arriving at that interface addressed to `192.168.0.5` would be ignored. However, at the same time, the `ppp0` interface may be assigned IP address `192.168.0.5`, and accept packets so addressed.

Different IP addresses on the same SIB brings us to the concept of routing. The SIB's kernel holds an internal routing table, which tells the network subsystem where to find remote computers. Specifically, the routing table holds zero or more entries stating which computers or "networks" may be found on a particular interface. For example, a single routing entry may specify that packets addressed to the computer with IP address `124.172.18.9` should be sent out over interface `sl0`.

A "network" is a group of computers with similar IP addresses. A network of computers may have one or more sections of their IP addresses in common. For example, computers with IP addresses 10.0.2.4 and 10.0.2.5 are on the same network. The number of sections the network machines have in common is determined by the "netmask". When the IP addresses and netmask are converted to binary, the netmask contains 1's in all locations where the IP addresses match. Zeros in the netmask may not appear to the left of 1's. For example, `10.0.2.5` and `10.0.2.78` are in a network with netmask `255.255.255.0` (this is called a "class C" network). Machines with addresses `127.6.83.9` and `127.4.14.62` would be on a network with netmask `255.0.0.0` (a "class A" network, very uncommon).

Each network also has a "broadcast address". The broadcast address is an IP address consisting of all of the sections of the IP which the machines have in common, and 0's in the rest. For example, in a network with machines numbered `192.168.*.*` and netmask `255.255.0.0`, the broadcast address is `192.168.0.0`. All machines on a network will receive a packet to the broadcast address.

It is possible (in fact, common) for an IP packet to be addressed to a machine which cannot be located in the SIB kernel's routing table. These packets are dealt with by means of a "default gateway." An SIB may have zero or one default gateway(s). The default gateway must be locatable by the kernel's routing table. Whenever a packet is encountered which is addressed to an unknown machine, it is sent to the default gateway. This gateway is almost always a switch, bridge, or router between one or more networks.

Each interface has an IP address, netmask, and broadcast address associated with it. This information may be set or viewed via the "`ifconfig`" command. With no parameters, `ifconfig` will display information for every active interface. Alternately, a particular interface may be specified on the command line, whose information will be display whether the interface is active or not (though the device must exist; the driver must be loaded). Interface configuration should be done through the master configuration utility (described later).

The kernel routing table may be changed or viewed using the "`route`" command. With no parameters, the route command will display the entire kernel routing table using hostnames instead of IP addresses where possible (hostnames are described in the next section). Each routing entry consists of a host or network address, a netmask, and an interface. The default gateway is an exception; this entry does not include an interface. The kernel must be able to locate the default gateway using the rest of the information in the routing table. Routing information should be changed through the master configuration script.

If your SIB does not use nameservers (described in the next section,) it is recommended that you use the "`-n`" command line switch with route. If you do not, the program will hang while it attempts to resolve IP addresses into hostnames, which will ultimately fail.

Some networks want to assign IP addresses and other interface information using a protocol named DHCP. The SIB includes a DHCP client, which is capable of automatically configuring the eth0 interface. You must have a DHCP server configured on your network to use DHCP. Use of DHCP should be specified when configuring the `eth0` interface with the master configuration script.

Name resolution

Internet servers are frequently referred to by a "hostname", rather than by IP address. The hostname takes the form of two or more strings of characters separated by dots, such as `www.emacinc.com`. The sections of the hostname generally have no correspondence to the sections of the IP address.

Internet-wide hostnames are registered with one of a few companies, usually for a fee. A registrant must also provide the IP address that a hostname is associated with. Hostname / IP address pairs are stored on DNS machines (Domain Name Service). DNS servers may run on machines with other services, such as mail and web server. Your company network and/or ISP should be able to provide you with a list of name servers. Note that name servers are always referred to by IP address; the paradox is obvious.

A hostname does not need to be internet-wide; every UNIX box has and uses a hostname, at least internally. The factory default hostname for the SIB is "`SIBx24.emacinc.com`" (no quotes.) The hostname of the SIB is set at boot time, using the hostname utility. Without arguments, this utility will also display the system's current hostname. The name set at boot time is the name contained in the `/etc/hostname` file. The contents of this file may be edited manually, of through the master configuration utility. Note that simply editing this file does not chance the hostname of a running system; the hostname utility must be used, or the system may be rebooted.

The SIB has two ways to resolve hostnames into IP addresses. The simplest is locally, via the `/etc/hosts` file. This file contains a local listing of hostname / IP address pairs which are accessible only to the SIB. This file always includes one entry, the `localhost / 127.0.0.1` pair. Generally, this file should also include a second entry relating the SIB's set hostname with the IP address of interface `eth0`. This entry is important, as some network applications will refuse to run if they cannot resolve the hostname of the SIB into the IP address of the interface they wish to use.

The second way the SIB can resolve hostnames into IP addresses is via DNS servers. Valid DNS servers should be listed in the `/etc/resolv.conf` file. Two to three DNS nameservers are common, but more are allowed. Note that if an application on the SIB attempts to resolve a hostname to an IP address via DNS servers, the application may stop for several minutes while the SIB attempts to contact the DNS server, especially if the DNS is inaccessible and the operation times out.

By default, the `/etc/hosts` file is searched before a DNS is accessed to resolve a name. This behavior may be altered by the `/etc/host.conf` file, which must be edited by hand. However, it is recommended that only an experienced Linux user do so.

Access control

Many common internet services (HTTP, FTP, TELNET) on the SIB are managed through a "superserver" called inetd. This superserver reads all incoming IP traffic, then passes the data to the appropriate service. Since all IP traffic passes through this single program, this allows for unified access control; the same configuration files can control access to many different internet services.

Control begins with the `/etc/services` files. This file contains a master list of all TCP and UDP ports that inetd will listen on, and the appropriate service to pass data on that port to. If a port is not

listed in this file, then inetd will ignore traffic on it (though other stand-alone network servers may listen on that port.) In general, this file should not be edited unless security is a major concern, and then only by those already knowledgeable about Linux.

The next line of defense are the files `/etc/hosts.allow` and `/etc/hosts.deny`. As their names suggests, these files allow a system administrator to explicitly list servers, services, users, networks, and domains which may or may not use the inetd services (a "domain" is the hostname equivalent of a network).

The `hosts.allow` file takes the greatest precedence. If an entity is granted access by the `hosts.allow` file, then access will be granted to that entity no matter what is listed in `hosts.deny`. The `hosts.deny` file lists entities that should be denied access to the SIB. If an entity appears in neither file, the default action is to allow access. By default, the SIB ships with a relatively permissive access policy. `Hosts.allow` and `hosts.deny` may be edited by hand should you wish to alter the access policy. EMAC recommends this be done only by an experienced Linux user. The format for these files is beyond the scope of this document.

The FTP server is configured by the files `/etc/ftpaccess`, `/etc/ftpusers`, `/etc/ftphosts`, and `/etc/ftpgroups`. Not all of these files are present on the SIB, though they may be added if desired. Ftpaccess is the main FTPD configuration file, the others are used for fine-grained access control. The format of these files is beyond the scope of this document. Configuration of the FTP server is accomplished by editing these files by hand. EMAC recommends this be done only by an experienced user.

PPP dial-in and -out

The same program works as both PPP server and client on the SIB. The `pppd` program may be invoked in one of two ways, both of which are simplified by scripts. The PPP protocol was designed to use dynamic IP addresses. This means that the PPP client (usually the machine doing the dialing) gets its IP address and interface configuration from the PPP server (usually the machine being dialed) before any IP packets are exchanged.

To use the SIB as a PPP client, you must first create a dial-out script, then execute it. Dial-out scripts should be created with the master configuration utility. These scripts are stored in the /etc/ppp/ directory. These scripts can be executed by typing the name of the script on the command line, just like any other program. These scripts may also be invoked by the `CRON` program, so that the SIB can connect to the internet via phone at specified times and upload collected data to a central server, for example.

PPP dial-out scripts will dial a phone number, enter a username and password, then optionally enter a command to start `ppp` on a remote machine. If further interaction with a remote server is required, an experienced Linux user may edit the `/etc/ppp/ppp-on-dialer` script.

The script which invokes `pppd` as a server is `/usr/bin/ppp`. This script will start the PPP server on any login terminal, including the console and tenet sessions. However, it is generally only useful when started on a dial-in terminal. To use the PPP server in the normal sense (as you would dial your ISP from home), you must first set the SIB's modem to answer the phone as a dial-in terminal. Once the dial-in terminal connection has been established, you must log in as a non-root user (usually `www`).After logging in, a PPP session may be started by simply typing "`ppp`" (no quotes) on the command line. This process of logging in and starting PPP is usually automated by dial scripts on the client machine. One PPP has been started, unreadable character strings will begin to appear in the dial-in terminal. This is PPP data;

at this point you should instruct the client machine to begin a PPP session.

When acting as a PPP server, the SIB needs two IP addresses assigned to a `ppp` interface before it is started – a local address which the SIB will use for the ppp interface, and a remote address to assign to the calling, client machine. For simplicity and logical reasons, these IP addresses are associated with serial devices rather than with network interfaces. These IP addresses are stored as `/etc/ppp/options.ttyS*`. Since the SIB's modem is on `ttyS2` by default, IP addresses will be stored as options.ttyS2. The format for these addresses is `<local IP>:<remote IP>`. An IP address of `0.0.0.0` means that PPP should try to get the address from the remote host, rather than assigning an IP address to the remote host.

Since these IP addresses are assigned to serial devices, and since the program works as both PPP server and client, pppd will use the addresses whether it is acting as a server or a client. That is, if `options.ttyS*` specifies IP addresses, then `pppd` will attempt to use those addresses even if it is dialing out. This can lead to problems if the remote computer is not expecting to have its IP address assigned to it. While this can be difficult to manage if a modem will be used for both PPP dial-in and dial-out, it also allows for a very useful feature: dial-on-demand.

Dial-on-demand is a feature whereby a `ppp` interface may be configured and activated, without dialing the phone. The interface will sit idle until the SIB's network subsystem is given a packet which must be transmitted to the remote host on the `ppp` interface which is using dial-on-demand. The SIB will then dial the number, make the connection, and transmit the data. After a certain amount of time without PPP traffic, the SIB will hang up the phone and again wait for outgoing PPP traffic.

To use PPP dial-on-demand, two things must be satisfied: first, a dial-out script must be created. Second, the modem device must have at least a remote IP address assigned to it. If the SIB does not know the remote address on the PPP link, it cannot know when to activate the link by dialing the modem. The local IP address may be specified, or it may be retrieved from the remote host after dialing. After creating the dial-out script in the master configuration utility, you will be given the option to make that script/link dial-on-demand. When this is done, the dial-out script will be run at boot time, and the `ppp` interface will be brought up. However, the phone will not dial until IP traffic addressed to the remote `ppp` IP address is encountered.

HTTP/Web server

The SIB's web server is lightweight, single-tasking HTTP server named "Boa" (`/usr/sbin/boa`). In testing, Boa has been shown to be quite fast, able to serve hundreds of requests per second on a 486-class machine. Since it is single-tasking, Boa also uses much less memory than some more standard HTTP servers (such as Apache). Boa also takes comparatively little flash space. The trade-off is that Boa does not have many of Apache's advanced features, such as ultra-fine-grained access control, or encryption. However, Boa has proven sufficient for most SIB tasks.

Boa is started at boot time by default. The script `/etc/init.d/netstd_init` starts both Boa (http) and the FTP server (ftpd). If this is not desired, it may be changed in two ways. First, the link S03netstd_init may be deleted from the `/etc/rc2.d` directory. This will prevent the `netstd_init` script from being executed when runlevel 2 (the default) is entered. Second, you may edit `the /etc/init.d/netstd_init` script by hand. Placing a '#' (no quotes) in front of the line which starts that http server will prevent the server from being started.

Boa's configuration is read from the `/etc/boa/boa.conf` file. This file contains directives for

TCP port, user and group to run as, logging, and other configurations. An HTTP server usually runs on port 80; it is not recommended that this be changed. Once the server starts, it runs as user "`nobody`" and group "`nogroup`." This is done for security reasons: the user `nobody` and group `nogroup` have no special permissions anywhere on the system. Therefore the HTTP server cannot read files that a non-privileged user should not, nor can it change file (via CGI, perhaps) that it should not.

To save RAM disk space, no logging is done. If you are having problems with the web server, logging may be enabled for both errors and HTTP accesses. The configuration file also allows you to set certain paths, directories, and file types. File types (MIME types) may be associated with file extensions both in the HTTP configuration file, and in the `/etc/mime.types` file. The paths tell the HTTP server where to find CGI and other files. In particular, Boa uses a CGI `/usr/lib/cgi-bin/boa_indexer`. This CGI generates directory listings on the fly when no index.html file is available. The `boa.conf` file is also somewhat self-documenting; see the comments in the file for more information.

One thing that cannot be set by the configuration file is the "server root." The server root is the directory that Boa considers the top-level directory; no higher levels can be accessed. For example, the default server root (compiled into Boa) is `/home/www`. When a browser is given the URL `http://10.0.2.41/` (assuming that the SIB is still at the default address `10.0.2.41`,) the contents of the `/home/www` directory will be displayed – there is no mechanism by which the `/home` or `/` directories can be displayed. Note, however, that Boa will follow directory links wherever they may lead, which could potentially include higher level directories than the server root. A non-default server root must be set on the command line; this can be done via the `/etc/init.d/httpd` script.

Note that Boa is a stand-alone server, which does not use the `inetd`. Therefore, inetd access control may not work when using Boa. Definitive information on Boa may be found at www.boa.org (as of this writing).

FTP server

The SIB's FTP server is the Linux-standard WU FTPD, version 2.6.1. This server is started by default at boot by the same scripts that begin the HTTP server, and it may be disabled in the same manner. WU-FTPD (`/usr/sbin/in.ftpd`) is run through `inetd`, thus all `inetd` access control methods will work as previously described. The file `/etc/ftpaccess` is the master ftpd configuration file. The files `/etc/ftphosts`, `/etc/ftpusers`, and `/etc/ftpgroups` all configure access control for the FTP server. The syntax of these files remains beyond the scope of this document; see the manual pages for these files on your development workstation for more detailed information. The SIB's FTP server uses the default ports 21 (for commands) and 20 (for data).

TELNET server

The SIB's TELNET server (`/usr/sbin/in.telnetd`) is also the Linux standard. The TELNET server is run through `inetd`, thus all access control methods previously described will work. Further access control is accomplished by the same methods as limiting terminal access to any user – the username / password pair, as well as the `/etc/securetty` file. See the manual pages on your development workstation for more details. The SIB's TELNET server runs on the default TELNET port 23.

Sending e-mail

To save space, the SIB uses a very small, simple program to transmit e-mail (reception of e-mail is not possible with the base SIB). The mail client is not a full-fledged mail transport agent; the SIB requires access to a properly configured SMTP mail server / transport agent. The setting up of such a machine is beyond the scope of this document. Your corporate network or local ISP probably have such a server, ask your system administrator for details. Usually, any machine on which you have an e-mail account will work.

The program to transmit mail is called "`smtpclient`." For a full list of options, type "`smtpclient –help`" on the command line. `Smtpclient` is a command-line based program, and as such all options such as subject and destination must be specified on the command line. At minimum, you will need to use the '-S' or "`–smtp-host=`" option to specify the IP address of the SMTP server, and a destination address (which does not require a command line switch). Depending on your mail server, the '`-f`' switch may also need to be used to specify who the message is from (some SMTP servers will not accept mail from users they do not recognize).

The `smtpclient` program reads its input from the "standard input." This means that you cannot use a switch to tell `smtpclient` the name of a file to send. The file must be passed by means of a "pipe" or by a "redirect." A pipe sends the output of one command to another. A pipe is specified by a '`|`' character. Use the cat command to print the desired file to the standard output, then use a '`|`' to pipe that output to the standard input of smtpclient: "`cat /tmp/mailfile | smtpclient -S mail.your.com -s Mail from SIB -f you@your.com destuser@dest.com`".

A redirect does something similar, but the file is specified after the smtpclient command line, and you don't need to cat the file. The command line "`smtpclient -S mail.your.com -s Mail from SIB -f you@your.com destuser@dest.com < /tmp/mailfile`" will do exactly the same thing as the command line using a pipe. Which you choose makes no difference. Mailing can also be done by a CRON task, or by a script. You may also write your own shell script to automate the mailing process.

Part II: Configuration

Getting started

The text-based SIB configuration program can be reached in one of three ways, either by telnet, the console, or by a serial terminal. To use telnet, plug the SIB ethernet interface into an ethernet port on a standard PC using a special point-to-point ethernet cable. Alternatively, the SIB and a standard PC could be plugged into an ethernet hub with no other devices attached.

The SIB is factory-configured to attempt DHCP configuration of its primary ethernet interface. If your network has a DHCP server, simply find out which address has been assigned to the SIB and TELNET to it. If The SIB fails to find a valid DHCP server, it will fall back to the factory defaults: IP address `10.0.2.41`, netmask `255.255.255.0`, broadcast `10.0.2.0`, default gateway `10.0.2.1` on interface `eth0`. (For future reference, these defaults are stored in `/etc/sysconfig/ethdefaults`.)

If this is the case, you will need to configure your PC to communicate with the SIB directly. Under Windows 95/98, this means changing your default gateway to `10.0.2.41` (the SIB's factory-set temporary IP address). Under GNU/Linux, this means adding a routing entry for `10.0.2.41` on the appropriate device (via the '`route`' command). You may then telnet to the SIB at address `10.0.2.41` using your standard telnet program.

To log in via serial terminal, a “dumb terminal” may be attached to the SIB's primary serial port (`COM1` or `ttyS0`). A standard PC running a terminal emulator such as Hyperterminal will also work. The terminal settings are 9600 bps, 8 data bits, 1 stop bit, no parity, no flow control, vt100 terminal emulation. Just plug in the terminal, power on the SIB, wait a minute or so for the SIB to complete its boot process, then press enter a few times to “wake up” the terminal and bring up a login prompt.

The “console” option may be used if the video option has been installed. The console is what might be considered the usual way to use a computer: plug a keyboard and VGA monitor into the appropriate connectors, turn the SIB on, and wait for a login prompt.

Once you are connected to the SIB, log in as “`root`”. If you are asked for a password, one should have been provided along with this documentation. If one has not been provided, try “`emac_inc`” (without quotes). As soon as the password is accepted, the main configuration menu should appear. This password should later be changed from the main configuration menu.

Main configuration menu

The main menu has eight options. '`Serial settings`' allows the user to configure a serial port as an IP interface (SLIP or PPP), a terminal interface (direct or dial-in), or a raw serial port. '`Ethernet settings`' sets up the information for the SIB's primary ethernet interface, including IP address. The '`Routing tables`' selection allows the user to add, delete, and view static routing information (i.e. which interfaces to use for particular destinations), including default gateway. '`Host name`' allows the user to change the SIB's hostname. '`Internet name resolution`' brings up the menu to configure DNS servers and manage static host entries. '`Edit gateway-cgi configuration file`' allows the user to directly edit the configuration file for the gateway-cgi program (discussed later). ‘`Configure periodic command scheduler`’ will bring up the contab configuration file in an editor, allowing CRON (discussed later) to be configured. '`Change configuration password`' allows the user to change the access password to configure the SIB (for security purposes, it is recommended that this be done first.) Finally, '`quit`' exits the program and bring up another login prompt. Each option is detailed here. Menu items marked with a “`*`” are those items necessary for almost every SIB configuration.

Serial settings*

Selecting “Serial settings” from the main menu will bring up a submenu, containing seven selections plus an option to return to the main menu. These seven options are:

Configure SLIP port

“Configure SLIP port” configures a specified serial device as an IP network interface, with its own IP address. Note that this IP address will only be used for this particular SLIP port; every IP network interface may have its own different IP address if desired. To configure a SLIP port, you first must choose a serial device which will be connected to a network. In Linux, these devices are called ttyS0 - ttyS3 (sometimes with “`/dev/`” prepended,) which correspond to DOS ports COM1 - COM4. You may select the device by number alone. Note that you are allowed to configure serial devices which do not actually exist in hardware -- this will not cause any system errors or stability issues, but errors will occur later if data is written to or read from these non-existent ports. Make sure you specify a valid device. After specification of a device, a screen will appear which contains the present SLIP configuration for the given port. If there is no information, that means that the port has not been previously configured for SLIP.

To complete configuration of a given SLIP device, you will need to specify at least a line speed and an IP address. If the SLIP device will be connected to only one other computer/ controller, 'Point to Point' should be specified. If the SLIP port will be connected to a network of other computers/controllers, do not specify point-to-point operation. If point-to-point is specified, you will be asked for the IP address of the machine to which the SLIP port will be attached. These IP addresses should be numeric address, not

hostnames. After you have entered the data, the new configuration will be displayed, and you will be asked if you wish to proceed. If you type 'y', the device will be configured as shown (and all other configuration for that device will be deleted.) If you type 'n', no configuration data will be changed, and the device will remain configured as originally shown. Whether the data is modified or not, you will automatically be returned to the serial configuration menu.

If multiple SLIP ports are configured, they should not share the same local IP address. For example, if both ttyS0 and ttyS1 are configured for SLIP, they should not both be given local IP address 192.168.1.77. This will cause confusion in the kernel routing tables. However, so long as each SLIP interface is allocated a separate IP address, as many SLIP interfaces as desired may be used.

Note that serial network interfaces are given names/numbers by the Linux kernel during system boot, in the order that the devices are configured. The first SLIP device configured is named "`sl0`," the second is "`sl1`," and so on. It is important to realize that these numbers are not necessarily equal to the serial port `ttySx` device numbers; `/dev/ttyS3` could be `sl0`, and so on. However, the `ttySx` devices are configured in numeric order; `ttyS0` is configured first, then `ttyS1`, and so on. Thus, the lowest numbered serial device configured as a SLIP port will be `sl0`, and so on. Thus if `ttyS0` is configured as a terminal port, `ttyS1` is a SLIP interface, `ttyS2` is a raw serial port, and `ttyS3` is a SLIP interface, `ttyS1` will be `sl0` and `ttyS3` will be `sl1`.

It should also be noted that in the serial section of the configuration program, serial devices are referred to by their device numbers (`ttySx`), not their interface numbers (`slx`). However, under the routing submenu (detailed later), SLIP ports are referred to by interface number.

Configure terminal port

'Configure terminal port' allows the user to set up a terminal server on the specified serial port. Serial devices are specified in the same manner as they were in 'Configuring SLIP ports' (as `ttySx`.) To set up a terminal server on a serial port, you will be asked for the line speed and the type of terminal to expect on the port. The terminal type typically starts with the letters '`vt`', common examples are "vt100" and "`vt320`". The "`vt100`" terminal type is the default, and is perhaps the most common terminal setting. After entering the new data, you will be shown the new configuration and asked for confirmation before any configuration data is changed. Again, accepting the configuration changes will delete any other configuration for the specified port. You will then be automatically returned to the serial configuration menu.

Configure dial-in terminal port

This menu item will allow the user to set up a terminal server that is modem-aware on the specified serial port. A serial device is specified as with previous menu items. When a serial port is configured this way and a modem is attached to it (or if the port is an internal modem), the SIB will answer the phone when it rings and attempt to make a data connection with the caller. If a data connection is made, the SIB will then prompt the caller for a login and password (in plain text), and the caller may log in as if they were using a standard terminal or the console.

The only parameter that must be specified is the line speed. As most modern modems are capable of high speed communications with the SIB serial port, this speed is 38400 by default. If the modem is internal, this parameter will by ignored by the hardware (though it is required as a placeholder for the software.) You will then be prompted for confirmation and returned to the serial configuration menu.

Configure dial-in PPP port

This item allows a user to set up a dial-in port to allow PPP connections from a remote client computer. Note that to use a port for dial-in PPP, that port must first be configured as a dial-in terminal port, as described previously. Three items are required to complete this setup: the serial tty (`ttySx`)

number where the modem is attached, a local IP address (the address the SIB will use for this PPP link), and a remote IP address (the address which the SIB PPP server will attempt to assign to the calling PPP client). If the calling client is to assign either or both IP addresses, then those addresses should be entered as `0.0.0.0`. This will tell the PPP server that the client is going to assign that particular address remotely.

To start a PPP link, dial into the SIB and log in as the **www** user. When presented with a command prompt, simply type "`ppp`" (no quotes). This will begin the PPP connection. Note that most PPP-aware dialing software attempts to automates this login process. Check the documentation for your client software for more information.

Note that configuration data may exist for a dial-in PPP port, and may be shown in "View serial settings," even after that port has been re-configured as something other than a dial-in terminal port. This is not an error; dial-in PPP configuration data will simply be ignored for any port that is not configured as a dial-in terminal port.

Configure dial-out PPP connection

This menu item will automatically generate a script which, when executed, will cause the SIB's modem (on a specified serial port) to dial and attempt to make a PPP connection with a remote host. However, instead of being referenced by device number as in other serial configurations, these dialer scripts are referenced by a free-form identifier string. This string is chosen and supplied by the user when creating the script. This allows the user to create and store multiple dialing configurations for the same modem on the same serial device.

Upon selection of this menu item, you will be prompted for an identifier string, a phone number, a username (on the remote machine), the password for that username, and an optional netmask. You will also be prompted for the command prompt on the remote machine, and the command the remote host requires to start PPP. These last two items are only necessary if you are required to start PPP manually on the remote machine. If the remote host automatically starts PPP upon login, simply leave these answers blank. If these are specified, the dialer program will wait until it receives the text you specify as the remote command prompt, then send the string you specify as the command to begin PPP. Note that if you specify punctuation or other special characters, they may not be displayed properly in the configuration program, and may cause trouble with the dialer script. This trouble can be avoided by entering an alphanumeric-only substring of the remote command prompt.

You will then be asked for confirmation before the script is generated. If you confirm this operation, a script with the name "`ppp-on-<identifier>`" will be generated in the `/bin` directory. Running this script will initiate a PPP link as specified. A general purpose "`ppp-off`" script may be used to terminate the PPP connection. Note that if there is more than one active PPP link (on more than one modem), you will need to specify the network device number (`ppp0`, `ppp1`, etc.) as a command line parameter to terminate the appropriate link; `ppp0` is the default. Network device numbers for PPP devices are assigned in the order that PPP connections are made; the first link to dial is `ppp0`, the second is `ppp1`, and so on.

Note that if you use dial-out PPP on a line which has a configuration file for dial-in PPP, the PPP daemon will attempt to use the dial-in IP data associated with the serial device when dialing out. For example, if you configured `ttyS3` with the PPP dial-in data `192.168.0.1:192.168.0.70` (local IP:remote IP), then generated a dial-out script for `ttyS3`, the PPP daemon would use `192.168.0.1` as the local IP address for the dial-out connection, and attempt to assign `192.168.0.70` to the machine being called during dial-out. If this is not desired, re-configure the dial-in data with the addresses "`0.0.0.0:0.0.0.0`". This will tell the PPP daemon to accept IP addresses from the machine being called (the PPP daemon will also accept IP addresses from the calling machine when answering a PPP

dial-in call on the specified `ttySx`.)

*Configure standard serial port**

'`Configure standard serial port`' will allow the user to set parameters such as line speed, number of data bits, and parity of a raw serial device at boot time. To configure a raw serial port, you will be prompted for a line speed, number of data bits, number of stop bits, parity type, and type of flow control. Line speed may be any standard line speed from 1200 to 38400 inclusive (note that high-speed UARTs often use their maximum speed when 38400 is specified; this may be 57k or 115k for newer UARTs). Data bits may be from 5 to 8, inclusive. Stop bits may be 1 or 2. Parity may be even, odd, mark, space, or none, specified by E, O, M, S, and N respectively. 'Mark' and 'Space' parity are for 9-bit (multidrop) networks. Flow control may be Xon/Xoff software, RTS/CTS hardware, RS-422, or none, specified by "`h`", "`s`", "`4`", and "`n`" respectively. If you use hardware flow control, make sure your cable has the appropriate number of wires. Note that some or all of these parameters may be overridden by user applications, including the `gateway-cgi` program.

If RS-422 is specified, then the RS-422 transmitter will be enabled at system boot, and will remain enabled during the entire operation of the SIB. Note that with standard SIB hardware, RS-422 is only available on port `ttyS1` (COM2). Make certain your RS-422 network is connected to the correct hardware connector before using this option.

You will be prompted for confirmation before any changes are made, then returned to the serial configuration menu. As with other configuration options, accepting changes will delete any SLIP or terminal configuration for the specified serial device.

View serial settings

'`View serial settings`' will allow the user to view all settings for a particular serial port. This function will check for terminal, SLIP, dial-up terminal, and raw configuration, as well as PPP dial-in configuration and any PPP dial-out scripts that use the specified port. Any existing configurations will be displayed one item at a time. Note that if there are multiple configurations, this is most likely an error, with the exception of PPP dial-in configurations, which will be ignored if the device is not configured as a dial-in terminal, and PPP dial-out scripts, which will only be used if explicitly executed. (Note, however, that PPP dial-out scripts should not be explicitly executed if the port is configured as something other than a raw serial device.)

Ethernet Settings*

The ethernet settings menu has only two items, 'configure ethernet device' and 'view network device settings.' Ethernet devices are assigned interface names/numbers by the Linux kernel at boot time, in the order in which they are configured by the boot process. The first ethernet device configured will be named "`eth0`," the second will be "`eth1`," and so on. Ethernet devices are referenced throughout the configuration program by these interface numbers.

*Configure ethernet device**

'Configure ethernet device' will prompt the user for a device (number only,) then bring up a screen with the current interface configuration data. The user will then be prompted you for new data. To complete configuration of an ethernet interface, you will need an IP address (numeric, in form A.B.C.D), a netmask (frequently `255.255.255.0`,) a broadcast address (frequently `A.B.C.255`,) and a network address (frequently `A.B.C.0`). After entering these parameters, you will be prompted for confirmation and returned to the main menu.

View network device settings

'View network device setting' will display the configuration information for a given interface. As with serial configurations, no data means that the specified device is not a configured ethernet interface.

Routing tables*

This submenu will allow the user to modify the static routing table, which the SIB uses to determine which network device to use when trying to communicate with a particular host. The submenu items are 'Add routing entry', 'Delete routing entry', 'Set default gateway', and 'View routing data.'

*Add routing entry**

Under '`Add routing data`,' you may add data for either a single host, or a network of hosts. At minimum, an SIB will need a host routing entry telling it on which network interface to find the default gateway (described in the next section). A single host will have a full address such as `10.0.2.1`, and a netmask is optional but not required. You must then specify the network device the SIB should use to reach this host. This will be the name of a network device such as `eth0` or `sl1`. Remember when attaching serial devices that the device (`ttySx`) number is not necessarily the interface (`slx`) number. However, boot initialization is set up such that the lowest number SLIP device will be given the interface number `sl0`. Thus if `ttyS0` and `ttyS1` are both raw serial devices and `ttyS2` is a SLIP port, `ttyS2` will be configured as `sl0`.

A network address will end in one or more zero fields, e.g. `192.168.0.0`. This will tell the SIB that this entry is valid for all destination addresses with the form `192.168.x.x`, where x can be anything from 1 to 254. For net entries, a netmask should be specified with zeros in the same locations as the zeros in the IP address, e.g. `255.255.0.0`. You must then specify the network interface the SIB should use to reach this range of addresses.

Before any routing data is modified, the routing table entry will be displayed and you will be prompted for confirmation. The routing entry will be in the form:

```
/sbin/route add -host (or -net) <IP number> netmask <netmask> dev <interface>
```

If you did not enter a netmask, the section "`netmask <netmask>`" will not appear in the entry. Press '`y`' to add the entry to the static routing table, or '`n`' to abort any modification.

Note that PPP routing entries will be added automatically when a connection is made. Therefore it is not possible to configure a routing entry for a PPP device, as the kernel will reject this route until the device exists (i.e. until the PPP daemon is running).

Delete routing entry

When 'Delete routing entry' is selected, a screen appears displaying the full current static routing table. This will consist of one or more entries of the form shown above, and one default routing entry of the form:

```
/sbin/route add default gw <IP address> metric 1
```

The meaning of the default route will be discussed in the following section. Below the routing table, you will be prompted for the IP address of the entry to delete. Note that entries will be matched by string comparison, so if you enter an address or other text common to more than one entry, then more than one entry will be selected for deletion. After you have entered the address to delete, you will be asked for confirmation, and any and all routes which would be deleted are displayed (this could be from none to all of the static routes, depending on the text you entered.) If you press '`y`', all of the routes displayed under the confirmation request will be deleted. Note that if no routes appear following the confirmation request, then none will be deleted regardless of your answer to the confirmation request.

Set default gateway*

The default gateway/route is where the SIB will send network packets when the destination of the packet has not been explicitly listed in the static routing table. The default gateway is generally your LAN's router, or other IP gatewaying/forwarding/switching device that will forward packets it receives onto another network. When you select the '`Set default gateway`' option, a screen will appear which lists the current default route. If no route information appears, then no default gateway has been set. You will then be prompted for the (numeric) IP address of the new default gateway. This new default gateway entry will then be displayed and you will be prompted for confirmation. Entering a '`y`' here will delete the previous default route and replace it with the one you have entered. Enter anything else and no changes will be made. Note that there should only be one default route in your routing table at any one time. Also note that prior to the definition of the default gateway, a static route to the gateway must be defined which includes a network device specification; if this entry does not exist, the networking subsystem will not know how to reach the default gateway.

View routing data

This option will display the full current routing table, including the default gateway. No modifications can be made from this screen. Pressing enter will return you to the routing table configuration submenu. Note that a standard host routing entry must be made for the default gateway prior to the specification of the default, or the SIB will not know how to reach the gateway.

Hostname

This option does not have a submenu. When selected, a screen will appear, stating the SIB's current hostname (this is pre-configured as `SIBx.emacinc.com`, and should be changed to suit your network.) You will then be prompted as to whether or not you wish to change the hostname. If no, you will be returned to the main menu. If yes, you will be prompted to enter a new hostname, then prompted for confirmation before any changes are made. Note that unless this hostname is properly registered with your network administrator (possibly your internet service provider), this hostname will be meaningless except to the SIB itself. Also note that the SIB will not be able to resolve its own hostname into its own IP address unless an appropriate static host entry is made under the 'Internet name resolution' submenu (described below.)

Internet name resolution

When the SIB networking subsystem is given a hostname, it must be resolved to a numeric IP address before any data can be sent. There are two ways in which a name may be resolved into an IP address. The first method is through static host entries. The SIB maintains a list of hostname/IP address pairs (each of which also has an optional alias). When given a hostname, the SIB first checks the hostname against this list, and if it finds a match, it uses the IP address found there. Static host entries may be managed by selecting '`Add static host entry`' and '`Delete static host entry`' from the name resolution submenu..

The second method by which a host name can be resolved into a numeric address is by communication with a name server. A name server is a computer which the SIB can access via network, which maintains an exhaustive list of host name / IP address entries. The SIB sends the host name it's looking for to the name server, and the name server sends a reply containing the numeric IP address of the host. Ask your network administrator or ISP for the IP addresses of any nameservers the SIB may have access to.

Add nameserver

When '`Add nameserver`' is selected, the current name server information will be displayed. This will consist of a single entry in the form "`search <domain>`" and one or more entries of the form

"nameserver <IP address>". It is common to have two or three name server entries. The <domain> in the search should contain the name of the domain in which the SIB will operate. If, for example, the EMAC computers were named emac0.emacinc.com through emac10.emacinc.com, the search entries would read "search emacinc.com".

Below the current nameserver information, you will be prompted as to whether you wish to change the search entry or add a nameserver (the default is add a nameserver). You will then be prompted for the domain name to search or the numeric IP address of a name server, respectively. The entry to be added to the name server configuration is then re-displayed, and you are asked for confirmation before changes are made. You will then be returned to the Internet name resolution submenu.

Delete nameserver

This submenu option will allow you to delete one or more entries from the current name server configuration. You will be prompted for the IP address of the entry to delete. As with the routing table, a string comparison is made, and as such multiple entries may be deleted at once (entering 'nameserver' will select all name server entries to be deleted.) After the address is entered, all lines to be deleted are displayed, and you are asked for confirmation before any modifications are made.

Add static host entry

This option allows you to associate a hostname with an IP address locally, so that the SIB will not need to contact a name server to resolve a host name. The full listing of static host entries will be displayed, and you will be prompted for a new hostname / IP address pair (with optional alias) to add. You should add entries for every remote host you wish to reference by name, but which do not have properly registered host names. You should also add entries for all hosts you wish to reference by name if you do not have access to a nameserver.

Delete static host entry

This option will allow you to delete an entry from the SIB's internal list of hostname / IP address pairs. The full listing will be displayed, and you will be prompted for the host name, IP address, or alias of the host to be deleted from the list. As always, you are prompted for confirmation before any changes are made.

View Internet name resolution configuration

This option will display both the list of host names / IP addresses and the name server configuration. Pressing enter will return you to the internet name resolution submenu.

Setting the time and date

Set the time and date by entering a string in the format 'mmddhhmmyyyy' (month, day, hour, minute, year). The time is set using a 24-hour system, and should reflect the current time zone.

Edit gateway-cgi configuration

This option has no function on an SIB without the gateway CGI.

Configure periodic command scheduler

This option allows the user to directly edit CRON's configuration file, /etc/crontab. The crontab file consists of zero of more entries with the basic format "at this time, execute this command". Since CRON executes on its own (with no user account configuration), crontab may also include statements which set and export the PATH shell variable (the default crontab file contains such statements,

which should not be modified except by an experienced user).

Times may be specified from the minute up to the month. The specific format for a crontab entry is:

```
<minute> <hour> <day of month> <month> <day of week> <user> <command>
```

All times may be specified by number, and the hour field goes by a 24-hour clock. The day of the week may be specified numerically, by the first three letters of its name. When specified numerically, the day of the week may be 0-7, with both 0 and 7 representing Sunday. Ranges and lists of times are supported. Thus, placing "5-10" in the hour field will cause the command to be executed every hour from five AM to ten AM. The equivalent (comma-separated) list would be "5, 6, 7, 8, 9, 10". Note that lists and ranges are not supported when using week day names. Divisors are also supported. For example, "5-10/2" in the hour column would execute the given command once every other hour between five and ten AM. Finally, "any" or "all" may be specified by placing a star ("*") in the field.

For example, the line

```
42  0  * 1,5  2-4 root /bin/echo "CRON command is executing" > /dev/console
```

Will print the message "CRON command is executing" to the SIB's console at 12:42 AM every Tuesday through Thursday during January and May. The field "user" contains the user that the command will be executed as. In most cases, this will be **root**. (Note that **root** has special privileges; care should be taken that essential system files are not modified by crontab entries.)

Crontab has several pre-defined entries, designed to perform standard system "housekeeping" tasks. These entries are designed to run every file (usually shell scripts) in a particular /etc subdirectory at regular intervals. These subdirectories are `cron.daily/`, `cron.weekly/`, and `cron.monthly/`. Simply put, everything in `/etc/cron.daily/` will be executed once per day (frequently late at night so as not to slow down user operations.) Everything in `/etc/cron.weekly/` will be executed once per week, and so on. This can be a good mechanism to limit the lengths of log files, and basic scripts to manage log files are already in the `CRON` directories. User files placed into these subdirectories will be executed along with the log file management scripts, providing a simple alternative to making actual crontab entries.

Shutdown SIB*

Choosing this option will allow the user to shut down the SIB, either for reboot or halt. As the SIB file system uses write caching, it requires proper shutdown to avoid potential loss of data. When this item is selected, you will be prompted as to whether this should be a halt or a reboot, with reboot being the default. When halt is selected, all software on the SIB will be stopped, the message "system halted" will appear, and the SIB will simply wait to be powered down. When reboot is selected, all SIB software will be halted, then a "warm" reboot sequence will be initiated, and the SIB will be restarted. As always, you will be prompted for confirmation before any action takes place.

Revisions

Rev 1.1: Modified to remove information on the gateway CGI

Rev 2.0: Major rewrite;

Revised introduction

Removed original “features” section

Added new, longer “features” section

Edited “configuration” section for modified software

Rev 2.4: Updated for new SIB software release.

Rev 2.4.2: Added tinylogin and busybox HTML page references.