new/usr/src/man/man1m/boot.1m 1 46017 Sat Jun 13 11:58:36 2015 new/usr/src/man/man1m/boot.1m 2056 update boot(1m) for right location of the GRUB menu \*\*\*\*\* 1 ′∖" te 2 .\" Copyright 2015 Nexenta Systems Inc. 3 .\" Copyright (c) 2008 Sun Microsystems, Inc. All Rights Reserved 4 .\" Copyright 1989 AT&T 5 .\" The contents of this file are subject to the terms of the Common Development 6 . \" See the License for the specific language governing permissions and limitati 7 . \" fields enclosed by brackets "[]" replaced with your own identifying informat 8 .TH BOOT 1M "Jun 7, 2015" 8 .TH BOOT 1M "Jan 14, 2015" 9 .SH NAME 10 boot \- start the system kernel or a standalone program 11 .SH SYNOPSIS 12 .SS "SPARC" 13 .LP 14 .nf 15 \fBboot\fR [\fIOBP\fR \fInames\fR] [\fIfile\fR] [\fB-aLV\fR] [\fB-F\fR \fIobject 16 [\fB-Z\fR \fIdataset\fR] [\fIboot-flags\fR] [\fB\(mi\(mi\fR] [\fIclient-pro 17 .fi 19 .SS "x86" 20 .LP 21 .nf 22 \fBkernel\$\fR \fB/platform/i86pc/kernel/\$ISADIR/unix\fR [\fIboot-args\fR] 23 [\fB-B\fR \fIprop\fR=\fIval\fR [,\fIval\fR...]] 24 .fi 26 .SH DESCRIPTION 27 .LP 28 Bootstrapping is the process of loading and executing a standalone program. For 29 the purpose of this discussion, bootstrapping means the process of loading and 30 executing the bootable operating system. Typically, the standalone program is 31 the operating system kernel (see fBkernel(R(1M)), but any standalone program 32 can be booted instead. On a SPARC-based system, the diagnostic monitor for a 33 machine is a good example of a standalone program other than the operating 34 system that can be booted. 35 .sp 36 .LP 37 If the standalone is identified as a dynamically-linked executable, \fBboot\fR 38 will load the interpreter (linker/loader) as indicated by the executable format 39 and then transfer control to the interpreter. If the standalone is 40 statically-linked, it will jump directly to the standalone. 41 .sp 42 .LP 43 Once the kernel is loaded, it starts the UNIX system, mounts the necessary file 44 systems (see fBvfstab(fR(4)), and runs fB/sbin/initfR to bring the system to 45 the "initdefault" state specified in fB/etc/inittabfR. See fBinittabfR(4). 46 .SS "SPARC Bootstrap Procedure" 47 .LP 48 On SPARC based systems, the bootstrap procedure on most machines consists of 49 the following basic phases. 50 .sp 51 .LP 52 After the machine is turned on, the system firmware (in PROM) executes power-on 53 self-test (POST). The form and scope of these tests depends on the version of 54 the firmware in your system. 55 .sp 56 .LP 57 After the tests have been completed successfully, the firmware attempts to 58 autoboot if the appropriate flag has been set in the non-volatile storage area 59 used by the firmware. The name of the file to load, and the device to load it 60 from can also be manipulated.

## new/usr/src/man/man1m/boot.1m

61 .sp 62 .LP 63 These flags and names can be set using the \fBeeprom\fR(1M) command from the 64 shell, or by using \fBPROM\fR commands from the \fBok\fR prompt after the 65 system has been halted. 66 .sp 68 The second level program is either a fileystem-specific boot block (when

67 T.P

69 booting from a disk), or \fBinetboot\fR or \fBwanboot\fR (when booting across 70 the network). 71 .sp 72 .LP 73 Network Booting 74 .sp 75 T.P 76 Network booting occurs in two steps: the client first obtains an IP address and 77 any other parameters necessary to permit it to load the second-stage booter. 78 The second-stage booter in turn loads the boot archive from the boot device. 79 .sp 80 .LP 81 An IP address can be obtained in one of three ways: RARP, DHCP, or manual 82 configuration, depending on the functions available in and configuration of the 83 PROM. Machines of the \fBsun4u\fR and \fBsun4v\fR kernel architectures have 84 DHCP-capable PROMs. 85 .sp 86 L.P. 87 The boot command syntax for specifying the two methods of network booting are: 88 .sp 89 .in +2 90 .nf 91 boot net:rarp 92 boot net:dhcp 93 .fi 94 .in -2 95 .sp 97 .sp 98 .LP 99 The command: 100 .sp 101 .in +2 102 .nf 103 boot net 104 .fi 105 .in -2 106 .sp 108 sp 109 .LP 110 without a \fBrarp\fR or \fBdhcp\fR specifier, invokes the default method for 111 network booting over the network interface for which \fBnet\fR is an alias. 112 .sp 113 .LP 114 The sequence of events for network booting using RARP/\fBbootparams\fR is 115 described in the following paragraphs. The sequence for DHCP follows the 116 RARP/\fBbootparams\fR description. 117 .sp 118 .LP

119 When booting over the network using RARP/\fBbootparams\fR, the PROM begins by 120 broadcasting a reverse ARP request until it receives a reply. When a reply is 121 received, the PROM then broadcasts a TFTP request to fetch the first block of 122 \fBinetboot\fR. Subsequent requests will be sent to the server that initially 123 answered the first block request. After loading, \fBinetboot\fR will also use 124 reverse ARP to fetch its IP address, then broadcast \fBbootparams\fR RPC calls

125 (see fBbootparams fR(4)) to locate configuration information and its root file 126 system. \fBinetboot\fR then loads the boot archive by means of NFS and

127 transfers control to that archive.

128 .sp

196 \fB\fBdhcp\fR\fR 130 When booting over the network using DHCP, the PROM broadcasts the hardware 197 .ad 131 address and kernel architecture and requests an IP address, boot parameters, 132 and network configuration information. After a DHCP server responds and is 198 .sp .6 133 selected (from among potentially multiple servers), that server sends to the 199 .RS 4n 134 client an IP address and all other information needed to boot the client. After 135 receipt of this information, the client PROM examines the name of the file to 201 .RE 136 be loaded, and will behave in one of two ways, depending on whether the file's 137 name appears to be an HTTP URL. If it does not, the PROM downloads 203 .sp 204 .ne 2 138 \fBinetboot\fR, loads that file into memory, and executes it. \fBinetboot\fR 139 loads the boot archive, which takes over the machine and releases 205 .na 206 \fB\fBprompt\fR\fR 140 \fBinetboot\fR. Startup scripts then initiate the DHCP agent (see 141 fBdhcpagentfR(1M), which implements further DHCP activities. 207 .ad 142 .sp 208 .sp .6 143 .LP 209 RS 4n 144 If the file to be loaded is an HTTP URL, the PROM will use HTTP to load the 145 referenced file. If the client has been configured with an HMAC SHA-1 key, it 211 .RE 146 will check the integrity of the loaded file before proceeding to execute it. 147 The file is expected to be the \fBwanboot\fR binary. The WAN boot process can 213 .sp 148 be configured to use either DHCP or NVRAM properties to discover the install 214 .ne 2 149 server and router and the proxies needed to connect to it. When \fBwanboot\fR 215 .na 216 \fB\fI<cmd>\fR\fR 150 begins executing, it determines whether sufficient information is available to 151 it to allow it to proceed. If any necessary information is missing, it will 217 .ad 152 either exit with an appropriate error or bring up a command interpreter and 218 .sp .6 153 prompt for further configuration information. Once \fBwanboot\fR has obtained 219 .RS 4n 154 the necessary information, it loads the boot loader into memory by means of 155 HTTP. If an encryption key has been installed on the client, \fBwanboot\fR will 221 .RE 156 verify the boot loader's signature and its accompanying hash. Presence of an 223 .sp 157 encryption key but no hashing key is an error. 158 .sp 224 .LP 159 .LP 160 The \fBwanboot\fR boot loader can communicate with the client using either HTTP 226 .sp 161 or secure HTTP. If the former, and if the client has been configured with an 227 LP 162 HMAC SHA-1 key, the boot loader will perform an integrity check of the root 163 file system. Once the root file system has been loaded into memory (and 229 .sp 164 possibly had an integrity check performed), the boot archive is transferred 230 .LP 165 from the server. If provided with a \fBboot\_logger\fR URL by means of the 166 \fBwanboot.conf\fR(4) file, \fBwanboot\fR will periodically log its progress. 167 .sp 168 .LP 169 Not all PROMs are capable of consuming URLs. You can determine whether a client 235 .sp 170 is so capable using the \fBlist-security-keys\fR OBP command (see 236 .ne 2 171 \fBmonitor\fR(1M)). 237 .na 238 \fB\fBhost-ip\fR\fR 172 .sp 173 .LP 239 .ad 174 WAN booting is not currently available on the x86 platform. 240 .sp .6 175 .sp 241 RS 4n 176 .LP 177 The \fBwanboot\fR Command Line 243 .RE 178 .sp 245 .sp 179 .LP 180 When the client program is \fBwanboot\fR, it accepts \fBclient-program-args\fR 246 .ne 2 247 .na 181 of the form: 182 .sp 248 \fB\fBrouter-ip\fR\fR 183 .in +2 249 .ad 184 .nf 250 .sp .6 185 boot ... -o \flopt1\fR[,\flopt2\fR[,...]] 251 .RS 4n 186 fi 187 .in -2 253 .RE 188 .sp 255 .sp 190 .sp 256 .ne 2 191 T.P 257 .na 192 where each option may be an action:

# new/usr/src/man/man1m/boot.1m

3

193 .sp 194 .ne 2 195 .na 200 Require \fBwanboot\fR to obtain configuration parameters by means of DHCP. 210 Cause \fBwanboot\fR to enter its command interpreter. 220 One of the interpreter commands listed below. 225 &... or an assignment, using the interpreter's parameter names listed below. 228 The \fBwanboot\fR Command Interpreter 231 The \fBwanboot\fR command interpreter is invoked by supplying a 232 \fBclient-program-args\fR of "\fB-o prompt\fR" when booting. Input consists of 233 single commands or assignments, or a comma-separated list of commands or 234 assignments. The configuration parameters are: 242 IP address of the client (in dotted-decimal notation) 252 IP address of the default router (in dotted-decimal notation) 258 \fB\fBsubnet-mask\fR\fR

new/usr/src/man/manlm/boot.lm	5	new/usr/src/man/manlm/boot.lm 6
259 .ad 260 .sp .6 261 .RS 4n 262 subnet mask (in dotted-decimal notation) 263 .RE		325 the HMAC SHA-1 signature key (40 hex ASCII characters) 326 .RE 328 .sp 329 .LP
265 .sp 266 .ne 2 267 .na 268 \fB\fBclient-id\fR\fR 269 .ad 270 .sp .6 271 .RS 4n 272 DHCP client identifier (a quoted ASCII string or hex ASCII) 273 .RE		<pre>330 Finally, the URL or the WAN boot CGI is referred to by means of: 331 .sp 332 .ne 2 333 .na 334 \fB\fBbootserver\fR\fR 335 .ad 336 .sp .6 337 .RS 4n 338 URL of WAN boot's CGI (the equivalent of OBP's \fBfile\fR parameter) 339 .RE</pre>
<pre>275 .sp 276 .ne 2 277 .na 278 \fB\fBhostname\fR\fR 279 .ad 280 .sp .6 281 .RS 4n 282 hostname to request in DHCP transactions (ASCII) 283 .RE 285 .sp 286 .ne 2</pre>		<pre>341 .sp 342 .LP 343 The interpreter accepts the following commands: 344 .sp 345 .ne 2 346 .na 347 \fB\fBhelp\fR\fR 348 .ad 349 .sp .6 350 .RS 4n 351 Print a brief description of the available commands 352 .RE</pre>
<pre>287 .na 288 \fB\fBhttp-proxy\fR\fR 289 .ad 290 .sp .6 291 .RS 4n 292 HTTP proxy server specification (IPADDR[:PORT]) 293 .RE 295 .sp 296 .LP 297 The key names are:</pre>		354 .sp 355 .ne 2 356 .na 357 \fB\fB\fIvar\fR=\fIval\fR\fR 358 .ad 359 .sp .6 360 .RS 4n 361 Assign \fIval\fR to \fIvar\fR, where \fIvar\fR is one of the configuration 362 parameter names, the key names, or \fBbootserver\fR. 363 .RE
298 .sp 299 .ne 300 .na 301 \fB\fB3des\fR\fR 302 .ad 303 .sp .6 304 .RS 4n 305 the triple DES encryption key (48 hex ASCII characters) 306 .RE		365 .sp 366 .ne 2 367 .na 368 \fB\fB\fIvar\fR=\fR 369 .ad 370 .sp .6 371 .RS 4n 372 Unset parameter \fIvar\fR. 373 .RE
<pre>308 .sp 309 .ne 2 310 .na 311 \fB\fBaes\fR 312 .ad 313 .sp .6 314 .RS 4n 315 the AES encryption key (32 hex ASCII characters) 316 .RE 318 .sp 319 .ne 2 320</pre>		<pre>375 .sp 376 .ne 2 377 .na 378 \fB\fBlist\fR\fR 379 .ad 380 .sp .6 381 .RS 4n 382 List all parameters and their values (key values retrieved by means of OBP are 383 never shown). 384 .RE</pre>
320 .na 321 \fBshal\fR\fR 322 .ad 323 .sp .6 324 .RS 4n		386 .sp 387 .ne 2 388 .na 389 \fB\fBprompt\fR\fR 390 .ad

7 391 .sp .6 392 .RS 4n 393 Prompt for values for unset parameters. The name of each parameter and its 394 current value (if any) is printed, and the user can accept this value (press 395 Return) or enter a new value. 396 RE 398 .sp 399 .ne 2 400 .na 401 \fB\fBqo\fR\fR 402 .ad 403 .sp .6 404 .RS 4n 405 Once the user is satisfied that all values have been entered, leave the 406 interpreter and continue booting. 407 RE 409 .sp 410 .ne 2 411 .na 412 \fB\fBexit\fR\fR 413 .ad 414 .sp .6 415 .RS 4n 416 Quit the boot interpreter and return to OBP's \fBok\fR prompt. 417 .RE 419 .sp 420 .LP 421 Any of these assignments or commands can be passed on the command line as part 422 of the \fB-o\fR options, subject to the OBP limit of 128 bytes for boot 423 arguments. For example, \fB-o\fR \fBlist,go\fR would simply list current 424 (default) values of the parameters and then continue booting. 425 .SS "iSCSI Boot" 426 .LP 427 iSCSI boot is currently supported only on x86. The host being booted must be 428 equipped with NIC(s) capable of iBFT (iSCSI Boot Firmware Table) or have the 429 mainboard's BIOS be iBFT-capable. iBFT, defined in the Advanced Configuration 430 and Power Interface (ACPI) 3.0b specification, specifies a block of information 431 that contains various parameters that are useful to the iSCSI Boot process. 432 .sp 433 .LP 434 Firmware implementing iBFT presents an iSCSI disk in the BIOS during startup as 435 a bootable device by establishing the connection to the iSCSI target. The rest 436 of the process of iSCSI booting is the same as booting from a local disk. 437 .sp 438 .LP 439 To configure the iBFT properly, users need to refer to the documentation from 440 their hardware vendors. 441 .SS "Booting from Disk" 442 .LP 443 When booting from disk, the OpenBoot PROM firmware reads the boot blocks from 444 blocks 1 to 15 of the partition specified as the boot device. This standalone 445 booter usually contains a file system-specific reader capable of reading the 446 boot archive. 447 .sp 448 .LP 449 If the pathname to the standalone is relative (does not begin with a slash). 450 the second level boot will look for the standalone in a platform-dependent 451 search path. This path is guaranteed to contain 452 \fB/platform/\fR\fIplatform-name\fR. Many SPARC platforms next search the 453 platform-specific path entry \fB/platform/\fR\fIhardware-class-name\fR. See 454 fBfilesystem fR(5). If the pathname is absolute, fBbootfR will use the 455 specified path. The fBbootfR program then loads the standalone at the

456 appropriate address, and then transfers control.

#### new/usr/src/man/man1m/boot.1m

457 .sp 458 .LP 459 Once the boot archive has been transferred from the boot device, Solaris can 460 initialize and take over control of the machine. This process is further 461 described in the "Boot Archive Phase," below, and is identical on all 462 platforms. 463 .sp 464 .LP 465 If the filename is not given on the command line or otherwise specified, for 466 example, by the \fBboot-file\fR NVRAM variable, \fBboot\fR chooses an 467 appropriate default file to load based on what software is installed on the 468 system and the capabilities of the hardware and firmware. 469 .sp 470 .LP 471 The path to the kernel must not contain any whitespace. 472 .SS "Booting from ZFS" 473 LP 474 Booting from ZFS differs from booting from UFS in that, with ZFS, a device 475 specifier identifies a storage pool, not a single root file system. A storage 476 pool can contain multiple bootable datasets (that is, root file systems). 477 Therefore, when booting from ZFS, it is not sufficient to specify a boot 478 device. One must also identify a root file system within the pool that was 479 identified by the boot device. By default, the dataset selected for booting is 480 the one identified by the pool's \fBbootfs\fR property. This default selection 481 can be overridden by specifying an alternate bootable dataset with the \fB-Z\fR 482 option. 483 .SS "Boot Archive Phase" 484 .LP 485 The boot archive contains a file system image that is mounted using an 486 in-memory disk. The image is self-describing, specifically containing a file 487 system reader in the boot block. This file system reader mounts and opens the 488 RAM disk image, then reads and executes the kernel contained within it. By 489 default, this kernel is in: 490 .sp 491 in +2 492 .nf 493 /platform/`uname -i`/kernel/unix 494 .fi 495 .in -2 496 .sp 498 .sp 499 .LP 500 If booting from ZFS, the pathnames of both the archive and the kernel file are 501 resolved in the root file system (that is, dataset) selected for booting as 502 described in the previous section. 503 .sp 504 .LP 505 The initialization of the kernel continues by loading necessary drivers and 506 modules from the in-memory filesystem until I/O can be turned on and the root 507 filesystem mounted. Once the root filesystem is mounted, the in-memory 508 filesystem is no longer needed and is discarded. 509 .SS "OpenBoot PROM \fBboot\fR Command Behavior" 510 .LP 511 The OpenBoot \fBboot\fR command takes arguments of the following form: 512 .sp 513 .in +2 514 .nf 515 ok boot [\fIdevice-specifier\fR] [\fIarguments\fR] 516 .fi 517 .in -2 518 .sp 520 .sp 521 LP 522 The default \fBboot\fR command has no arguments:

9 523 .sp 524 .in +2 525 .nf 526 ok boot 527 .fi 528 .in -2 529 .sp 531 .sp 532 .LP 533 If no \fIdevice-specifier\fR is given on the \fBboot\fR command line, OpenBoot 534 typically uses the \flboot-device\fR or \fldiag-device\fR \fBNVRAM\fR variable. 535 If no optional \flarguments\fR are given on the command line, OpenBoot 536 typically uses the \fIboot-file\fR or \fIdiag-file\fR \fBNVRAM\fR variable as 537 default \fBboot\fR arguments. (If the system is in diagnostics mode, 538 \fIdiag-device\fR and \fIdiag-file\fR are used instead of \fIboot-device\fR and 539 \flboot-file\fR). 540 .sp 541 .LP 542 \flarguments\fR may include more than one string. All \flargument\fR strings 543 are passed to the secondary booter; they are not interpreted by OpenBoot. 544 .sp 545 .LP 546 If any \flarguments\fR are specified on the \fBboot\fR command line, then 547 neither the \flboot-file\fR nor the \fldiag-file\fR \fBNVRAM\fR variable is 548 used. The contents of the \fBNVRAM\fR variables are not merged with command 549 line arguments. For example, the command: 550 .sp 551 .in +2 552 .nf 553 ok \fBboot\fR \fB-s\fR 554 .fi 555 .in -2 556 .sp 558 .sp 559 .LP 560 ignores the settings in both \fIboot-file\fR and \fIdiag-file\fR; it interprets 561 the string \fB"-s"\fR as \fIarguments\fR. \fBboot\fR will not use the contents 562 of \flboot-file\fR or \fldiag-file\fR. 563 .sp 564 .LP 565 With older PROMs, the command: 566 .sp 567 .in +2 568 .nf 569 ok \fBboot net\fR 570 .fi 571 .in -2 572 .sp 574 .sp 575 .LP 576 took no arguments, using instead the settings in \fIboot-file\fR or 577 \fIdiag-file\fR (if set) as the default file name and arguments to pass to 578 boot. In most cases, it is best to allow the  $fBboot\fR$  command to choose an 579 appropriate default based upon the system type, system hardware and firmware, 580 and upon what is installed on the root file system. Changing \flboot-file\fR or 581 \fIdiag-file\fR can generate unexpected results in certain circumstances. 582 .sp 583 .LP 584 This behavior is found on most OpenBoot 2.x and 3.x based systems. Note that 585 differences may occur on some platforms. 586 .sp 587 .LP 588 The command:

#### new/usr/src/man/man1m/boot.1m

589 .sp 590 .LP

- 591 ok \fBboot cdrom\fR

592 .sp 593 .LP 594 \&...also normally takes no arguments. Accordingly, if \fIboot-file\fR is set 595 to the 64-bit kernel filename and you attempt to boot the installation CD or 596 DVD with \fBboot cdrom\fR, boot will fail if the installation media contains 597 only a 32-bit kernel. 598 .sp 599 .LP 600 Because the contents of \fIboot-file\fR or \fIdiag-file\fR can be ignored 601 depending on the form of the \fBboot\fR command used, reliance upon 602 \fiboot-file\fR should be discouraged for most production systems. 603 .sp 604 .LP 605 When executing a WAN boot from a local (CD or DVD) copy of wanboot, one must 606 use: 607 .sp 608 .LP 609 ok \fBboot cdrom -F wanboot - install\fR 610 .sp 611 .LP 612 Modern PROMs have enhanced the network boot support package to support the 613 following syntax for arguments to be processed by the package: 614 .sp 615 .LP 616 [\fIprotocol\fR,] [\fIkey\fR=\fIvalue\fR,]\* 617 .sp 618 .LP 619 All arguments are optional and can appear in any order. Commas are required 620 unless the argument is at the end of the list. If specified, an argument takes 621 precedence over any default values, or, if booting using DHCP, over 622 configuration information provided by a DHCP server for those parameters. 623 .sp 624 .LP 625 \fIprotocol\fR, above, specifies the address discovery protocol to be used. 626 .sp 627 .LP 628 Configuration parameters, listed below, are specified as \fIkey\fR=\fIvalue\fR 629 attribute pairs. 630 .sp 631 .ne 2 632 .na 633 \fB\fBtftp-server\fR\fR 634 ad 635 .sp .6 636 .RS 4n 637 IP address of the TFTP server 638 .RE 640 .sp 641 .ne 2 642 .na 643 \fB\fBfile\fR\fR 644 .ad 645 .sp .6 646 .RS 4n 647 file to download using TFTP or URL for WAN boot 648 .RE 650 .sp 651 .ne 2 652 .na 653 \fB\fBhost-ip\fR\fR 654 .ad

new/usr/src/man/manlm/boot.lm	11	new/usr/src/man/manlm/boot.lm	12
655 .sp .6 656 .RS 4n 657 IP address of the client (in dotted-decimal notation) 658 .RE		721 .ne 2 722 .na 723 \fB\fBdhcp-retries\fR\fR 724 .ad 725 .sp .6	
<pre>660 .sp 661 .ne 2 662 .na 663 \fB\fBrouter-ip\fR\fR 664 .ad</pre>		726 .RS 4n 727 maximum number of DHCP retries 728 .RE 730 .sp	
665 .sp .6 666 .RS 4n 667 IP address of the default router 668 .RE		731 .LP 732 The list of arguments to be processed by the network boot support package is 733 specified in one of two ways: 734 .RS +4 735 .TP	
<pre>670 .sp 671 .ne 2 672 .na 673 \fB\fBsubnet-mask\fR\fR 674 .ad 675 .sp .6 676 .RS 4n</pre>		<pre>736 .ie t \(bu 737 .el o 738 As arguments passed to the package's \fBopen\fR method, or 739 .RE 740 .RS +4 741 .TP 742 .ie t \(bu</pre>	
677 subnet mask (in dotted-decimal notation) 678 .RE 680 .sp 681 .ne 2		743 .el o 744 arguments listed in the NVRAM variable \fBnetwork-boot-arguments\fR. 745 .RE 746 .sp 747 .LP	
<pre>682 .na 683 \fB\fBclient-id\fR\fR 684 .ad 685 .sp .6 686 .RS 4n 687 DHCP client identifier</pre>		<pre>748 Arguments specified in \fBnetwork-boot-arguments\fR will be processed only if 749 there are no arguments passed to the package's \fBopen\fR method. 750 .sp 751 .LP 752 Argument Values 753 .sp</pre>	
688 .RE 690 .sp 691 .ne 2 692 .na		754 .LP 755 \fIprotocol\fR specifies the address discovery protocol to be used. If present 756 the possible values are \fBrarp\fR or \fBdhcp\fR. 757 .sp 758 .LP	t,
693 \fB\fBhostname\fR\fR 694 .ad 695 .sp .6 696 .RS 4n 697 hostname to use in DHCP transactions		759 If other configuration parameters are specified in the new syntax and style 760 specified by this document, absence of the \fIprotocol\fR parameter implies 761 manual configuration. 762 .sp 763 .LP	
698 .RE 700 .sp 701 .ne 2		764 If no other configuration parameters are specified, or if those arguments are 765 specified in the positional parameter syntax currently supported, the absence 766 of the \fIprotocol\fR parameter causes the network boot support package to use 767 the platform-specific default address discovery protocol.	
702 .na 703 \fB\fBhttp-proxy\fR\fR 704 .ad 705 .sp .6 706 .RS 4n 707 HTTP proxy server specification (IPADDR[:PORT]) 708 .RE		768 .sp 769 .LP 770 Manual configuration requires that the client be provided its IP address, the 771 name of the boot file, and the address of the server providing the boot file 772 image. Depending on the network configuration, it might be required that 773 \fBsubnet-mask\fR and \fBrouter-ip\fR also be specified. 774 .sp	
<pre>710 .sp 711 .ne 2 712 .na 713 \fB\fBtftp-retries\fR\fR 714 .ad 715 .sp .6 716 .RS 4n 717 maximum number of TFTP retries</pre>		775 .LP 776 ILP 776 If the \fIprotocol\fR argument is not specified, the network boot support 777 package uses the platform-specific default address discovery protocol. 778 .sp 779 .LP 780 \fBftfp-server\fR is the IP address (in standard IPv4 dotted-decimal notation 781 of the TFTP server that provides the file to download if using TFTP. 782 .sp 783 .LP	)
718 .RE 720 .sp		784 When using DHCP, the value, if specified, overrides the value of the TFTP 785 server specified in the DHCP response. 786 .sp	

13 787 LP 788 The TFTP RRO is unicast to the server if one is specified as an argument or in 789 the DHCP response. Otherwise, the TFTP RRQ is broadcast. 790 .sp 791 .LP 792 \fIfile\fR specifies the file to be loaded by TFTP from the TFTP server, or the 793 URL if using HTTP. The use of HTTP is triggered if the file name is a URL, that 794 is, the file name starts with \fBhttp:\fR (case-insensitive). 795 .sp 796 .LP 797 When using RARP and TFTP, the default file name is the ASCII hexadecimal 798 representation of the IP address of the client, as documented in a preceding 799 section of this document. 800 .sp 801 LP 802 When using DHCP, this argument, if specified, overrides the name of the boot 803 file specified in the DHCP response. 804 .sp 805 .LP 806 When using DHCP and TFTP, the default file name is constructed from the root 807 node's \fBname\fR property, with commas (,) replaced by periods (.). 808 .sp 809 .LP 810 When specified on the command line, the filename must not contain slashes 811 (\fB/\fR). 812 .sp 813 .LP 814 The format of URLs is described in RFC 2396. The HTTP server must be specified 815 as an IP address (in standard IPv4 dotted-decimal notation). The optional port 816 number is specified in decimal. If a port is not specified, port 80 (decimal) 817 is implied. 818 .sp 819 .LP 820 The URL presented must be "safe-encoded", that is, the package does not apply 821 escape encodings to the URL presented. URLs containing commas must be presented 822 as a quoted string. Quoting URLs is optional otherwise. 823 .sp 824 .LP 825 \fBhost-ip\fR specifies the IP address (in standard IPv4 dotted-decimal 826 notation) of the client, the system being booted. If using RARP as the address 827 discovery protocol, specifying this argument makes use of RARP unnecessary. 828 .sp 829 .LP 830 If DHCP is used, specifying the \fBhost-ip\fR argument causes the client to 831 follow the steps required of a client with an "Externally Configured Network 832 Address", as specified in RFC 2131. 833 .sp 834 .LP 835 \fBrouter-ip\fR is the IP address (in standard IPv4 dotted-decimal notation) of 836 a router on a directly connected network. The router will be used as the first 837 hop for communications spanning networks. If this argument is supplied, the 838 router specified here takes precedence over the preferred router specified in 839 the DHCP response. 840 .sp 841 .LP 842 \fBsubnet-mask\fR (specified in standard IPv4 dotted-decimal notation) is the 843 subnet mask on the client's network. If the subnet mask is not provided (either 844 by means of this argument or in the DHCP response), the default mask 845 appropriate to the network class (Class A, B, or C) of the address assigned to 846 the booting client will be assumed. 847 .sp 848 LP 849 \fBclient-id\fR specifies the unique identifier for the client. The DHCP client 850 identifier is derived from this value. Client identifiers can be specified as: 851 .RS +4

852 .TP

new/usr/src/man/man1m/boot.1m

853 .ie t \(bu 854 .el o 855 The ASCII hexadecimal representation of the identifier, or 856 .RE 857 .RS +4 858 .TP 859 .ie t \(bu 860 .el o 861 a guoted string 862 .RE 863 .sp 864 .LP 865 Thus, \fBclient-id="openboot"\fR and \fBclient-id=6f70656e626f6f74\fR both 866 represent a DHCP client identifier of 6F70656E626F6F74. 867 .sp 868 .LP 869 Identifiers specified on the command line must must not include slash (fB/fR)870 or spaces. 871 .sp 872 .LP 873 The maximum length of the DHCP client identifier is 32 bytes, or 64 characters 874 representing 32 bytes if using the ASCII hexadecimal form. If the latter form 875 is used, the number of characters in the identifier must be an even number. 876 Valid characters are 0-9, a-f, and A-F. 877 .sp 878 .LP 879 For correct identification of clients, the client identifier must be unique 880 among the client identifiers used on the subnet to which the client is 881 attached. System administrators are responsible for choosing identifiers that 882 meet this requirement. 883 .sp 884 .LP 885 Specifying a client identifier on a command line takes precedence over any 886 other DHCP mechanism of specifying identifiers. 887 .sp 888 .LP 889 \fBhostname\fR (specified as a string) specifies the hostname to be used in 890 DHCP transactions. The name might or might not be qualified with the local 891 domain name. The maximum length of the hostname is 255 characters. 892 T.P 893 Note -894 .sp 895 .RS 2 896 The \fBhostname\fR parameter can be used in service environments that require 897 that the client provide the desired hostname to the DHCP server. Clients 898 provide the desired hostname to the DHCP server, which can then register the 899 hostname and IP address assigned to the client with DNS. 900 .RE 901 .sp 902 .LP 903 \fBhttp-proxy\fR is specified in the following standard notation for a host: 904 .sp 905 .in +2 906 .nf 907 \flhost\fR [":"" \flport\fR] 908 .fi 909 .in -2 910 .sp 912 .sp 913 .LP 914 \&...where \flhost\fR is specified as an IP ddress (in standard IPv4 915 dotted-decimal notation) and the optional \fIport\fR is specified in decimal. 916 If a port is not specified, port 8080 (decimal) is implied. 917 .sp 918 .LP

15

919 \fBtftp-retries\fR is the maximum number of retries (specified in decimal) 920 attempted before the TFTP process is determined to have failed. Defaults to

- 921 using infinite retries.
- 922 .sp

923 .LP

924 \fBdhcp-retries\fR is the maximum number of retries (specified in decimal) 925 attempted before the DHCP process is determined to have failed. Defaults to of 926 using infinite retries.

- 927 .SS "x86 Bootstrap Procedure"
- 928 .LP

929 On x86 based systems, the bootstrapping process consists of two conceptually 930 distinct phases, kernel loading and kernel initialization. Kernel loading is 931 implemented in GRUB (GRand Unified Bootloader) using the BIOS ROM on the system 932 board, and BIOS extensions in ROMs on peripheral boards. The BIOS loads GRUB, 933 starting with the first physical sector from a hard disk, DVD, or CD. If 934 supported by the ROM on the network adapter, the BIOS can also download the 935 \fBpxegrub\fR binary from a network boot server. Once GRUB is located, it 936 executes a command in a menu to load the \fBunix\fR kernel and a

937 pre-constructed boot archive containing kernel modules and data.

938 .sp

939 .LP

940 If the device identified by GRUB as the boot device contains a ZFS storage 941 pool, the \fBmenu.lst\fR file used to create the GRUB menu will be found in the 942 dataset at the root of the pool's dataset hierarchy. This is the dataset with 943 the same name as the pool itself. There is always exactly one such dataset in a 944 pool, and so this dataset is well-suited for pool-wide data such as the

- 945 \fBmenu.lst\fR file. After the system is boted, this dataset is mounted at
- 946 /\flpoolname\fR in the root file system.

947 .sp

948 .LP 949 There can be multiple bootable datasets (that is, root file systems) within a 950 pool. By default, the file system in which file name entries in a 951 \fBmenu.lst\fR file are resolved is the one identified by the pool's 952 \fBbootfs\fR property (see \fBzpool\fR(1M)). However, a \fBmenu.lst\fR entry 953 can contain a \fBbootfs\fR command, which specifies an alternate dataset in the 954 pool. In this way, the \fBmenu.lst\fR file can contain entries for multiple 955 root file systems within the pool. 956 .sp 957 .LP

958 Kernel initialization starts when GRUB finishes loading the boot archive and 959 hands control over to the \fBunix\fR binary. At this point, GRUB becomes 960 inactive and no more I/O occurs with the boot device. The Unix operating system 961 initializes, links in the necessary modules from the boot archive and mounts 962 the root file system on the real root device. At this point, the kernel regains 963 storage I/O, mounts additional file systems (see \fBvfstab\fR(4)), and starts 964 various operating system services (see \fBsmf\fR(5)).

965 .SS "Failsafe Mode"

966 .LP

967 A requirement of booting from a root filesystem image built into a boot archive 968 then remounting root onto the actual root device is that the contents of the 969 boot archive and the root filesystem must be consistent. Otherwise, the proper 970 operation and integrity of the machine cannot be guaranteed. 971 .sp

972 .LP

973 The term "consistent" means that all files and modules in the root filesystem 974 are also present in the boot archive and have identical contents. Since the 975 boot strategy requires first reading and mounting the boot archive as the 976 first-stage root image, all unloadable kernel modules and initialization 977 derived from the contents of the boot archive are required to match the real 978 root filesystem. Without such consistency, it is possible that the system could 979 be running with a kernel module or parameter setting applied to the root device 980 before reboot, but not yet updated in the root archive. This inconsistency 981 could result in system instability or data loss.

982 .sp 983 LP

984 Once the root filesystem is mounted, and before relinquishing the in-memory

# new/usr/src/man/man1m/boot.1m

985 filesystem, Solaris performs a consistency verification against the two file 986 systems. If an inconsistency is detected, Solaris suspends the normal boot 987 sequence and falls back to failsafe mode. Correcting this state requires the 988 administrator take one of two steps. The recommended procedure is to reboot to 989 the failsafe archive and rebuild the boot archive. This ensures that a known 990 kernel is booted and functioning for the archive rebuild process. 991 Alternatively, the administrator can elect to clear the inconsistent boot 992 archive service state and continue system bring-up if the inconsistency is such 993 that correct system operation will not be impaired. See \fBsvcadm\fR(1M). 994 .sp 995 .LP 996 If the boot archive service is cleared and system bring-up is continued (the 997 second alternative above), the system may be running with unloadable kernel 998 drivers or other modules that are out-of-date with respect to the root 999 filesystem. As such, correct system operation may be compromised. 1000 .sp 1001 LP 1002 To ensure that the boot archive is consistent, the normal system shutdown 1003 process, as initiated by fBreboot fR(1M) and fBshutdown fR(1M), checks for 1004 and applies updates to the boot archive at the conclusion of the 1005 \fBumountall\fR(1M) milestone. 1006 .sp 1007 .LP 1008 An update to any kernel file, driver, module or driver configuration file that 1009 needs to be included in the boot archive after the \fBumountall\fR service is 1010 complete will result in a failed boot archive consistency check during the next 1011 boot. To avoid this, it is recommended to always shut down a machine cleanly. 1012 .sp 1013 .LP 1014 If an update is required to the kernel after completion of the \fBumountall\fR 1015 service, the administrator may elect to rebuild the archive by invoking: 1016 .sp 1017 .in +2 1018 .nf 1019 # \fBbootadm update-archive\fR 1020 .fi 1021 .in -2 1022 .sp 1024 .SS "Failsafe Boot Archive" 1025 .LP 1026 The failsafe archive can be used to boot the machine at any time for 1027 maintenance or troubleshooting. The failsafe boot archive is installed on the 1028 machine, sourced from the miniroot archive. Booting the failsafe archive causes 1029 the machine to boot using the in-memory filesystem as the root device. 1030 .SS "SPARC" 1031 .LP 1032 The SPARC failsafe archive is: 1033 .sp 1034 .in +2 1035 nf 1036 /platform/`uname -i`/failsafe 1037 .fi 1038 .in -2 1039 .sp 1041 .sp 1042 .LP 1043 \&...and can be booted as follows: 1044 .sp 1045 .in +2 1046 .nf 1047 ok \fBboot [\fIdevice-specifier\fR] -F failsafe\fR 1048 .fi

- 1049 .in -2
- 1050 .sp

1052 .sp 1053 .LP 1054 If a user wishes to boot a failsafe archive from a particular ZFS bootable 1055 dataset, this can be done as follows: 1056 .sp 1057 .in +2 1058 .nf 1059 ok \fBboot [\fIdevice-specifier\fR] -Z \fIdataset\fR -F failsafe\fR 1060 .fi 1061 .in -2 1062 .sp 1064 .SS "x86" 1065 .LP 1066 The x86 failsafe archive is: 1067 .sp 1068 .in +2 1069 .nf 1070 /boot/x86.miniroot-safe 1071 .fi 1072 .in -2 1073 .sp 1075 .sp 1076 .LP 1077 \&...and can be booted by selecting the \fBSolaris failsafe\fR item from the 1078 GRUB menu. 1079 .SH OPTIONS 1080 .SS "SPARC" 1081 .LP 1082 The following SPARC options are supported: 1083 .sp 1084 .ne 2 1085 .na 1086 \fB\fB-a\fR\fR 1087 .ad 1088 .sp .6 1089 .RS 4n 1090 The boot program interprets this flag to mean \fBask me\fR, and so it prompts 1091 for the name of the standalone. The  $fB\&'\fR = a\fR & fR\fB & r$ 1092 passed to the standalone program. 1093 .RE 1095 .sp 1096 .ne 2 1097 .na 1098 \fB\fB-D\fR \fIdefault-file\fR\fR 1099 .ad 1100 .sp .6 1101 .RS 4n 1102 Explicitly specify the \fldefault-file\fR. On some systems, \fBboot\fR chooses 1103 a dynamic default file, used when none is otherwise specified. This option 1104 allows the \fIdefault-file\fR to be explicitly set and can be useful when 1105 booting \fBkmdb\fR(1) since, by default, \fBkmdb\fR loads the default-file as 1106 exported by the \fBboot\fR program. 1107 .RE 1109 .sp 1110 .ne 2 1111 .na 1112 \fB\fB-F\fR \fIobject\fR\fR 1113 .ad 1114 .sp .6 1115 .RS 4n 1116 Boot using the named object. The object must be either an ELF executable or

17

new/usr/src/man/man1m/boot.1m 18 1117 bootable object containing a boot block. The primary use is to boot the 1118 failsafe or \fBwanboot\fR boot archive. 1119 .RE 1121 .sp 1122 .ne 2 1123 .na 1124 \fB\fB-L\fR\fR 1125 .ad 1126 .sp .6 1127 .RS 4n 1128 List the bootable datasets within a ZFS pool. You can select one of the 1129 bootable datasets in the list, after which detailed instructions for booting 1130 that dataset are displayed. Boot the selected dataset by following the 1131 instructions. This option is supported only when the boot device contains a ZFS 1132 storage pool. 1133 .RE 1135 .sp 1136 .ne 2 1137 .na 1138 \fB\fB-V\fR\fR 1139 .ad 1140 .sp .6 1141 .RS 4n 1142 Display verbose debugging information. 1143 .RE 1145 .sp 1146 .ne 2 1147 .na 1148 \fB\fIboot-flags\fR\fR 1149 .ad 1150 .sp .6 1151 RS 4n 1152 The boot program passes all \fIboot-flags\fR to \fBfile\fR. They are not 1153 interpreted by fBboot R. See the fBkernel R(1M) and fBkmdb R(1) manual 1154 pages for information about the options available with the default standalone 1155 program. 1156 .RE 1158 .sp 1159 .ne 2 1160 .na 1161 \fB\fIclient-program-args\fR\fR 1162 ad 1163 .sp .6 1164 .RS 4n 1165 The \fBboot\fR program passes all \fIclient-program-args\fR to \fIfile\fR. They 1166 are not interpreted by \fBboot\fR. 1167 .RE 1169 .sp 1170 .ne 2 1171 .na 1172 \fB\fIfile\fR\fR 1173 .ad 1174 .sp .6 1175 .RS 4n 1176 Name of a standalone program to \fBboot\fR. If a filename is not explicitly 1177 specified, either on the fBbootfR command line or in the fIboot-fileR1178 NVRAM variable, \fBboot\fR chooses an appropriate default filename. 1179 .RE 1181 .sp 1182 .ne 2

1183 .na 1184 \fB\fIOBP\fR \fInames\fR\fR 1185 .ad 1186 .sp .6 1187 .RS 4n 1188 Specify the open boot prom designations. For example, on Desktop SPARC based 1189 systems, the designation \fB/sbus/esp@0.800000/sd@3.0:a\fR indicates a 1190 \fBSCSI\fR disk (sd) at target 3, lun0 on the \fBSCSI\fR bus, with the esp host 1191 adapter plugged into slot 0. 1192 .RE 1194 .sp 1195 .ne 2 1196 .na 1197 \fB\fB-Z\fR \fIdataset\fR\fR 1198 .ad 1199 .sp .6 1200 .RS 4n 1201 Boot from the root file system in the specified ZFS dataset. 1202 RE 1204 .SS "x86" 1205 .LP 1206 The following x86 options are supported: 1207 .sp 1208 .ne 2 1209 .na 1210 \fB\fB-B\fR \fIprop\fR=\fIval\fR...\fR 1211 .ad 1212 .sp .6 1213 .RS 4n 1214 One or more property-value pairs to be passed to the kernel. Multiple 1215 property-value pairs must be separated by a comma. Use of this option is the 1216 equivalent of the command: \fBeeprom\fR \fIprop\fR=\fIval\fR. See 1217 \fBeeprom\fR(1M) for available properties and valid values. 1218 .sp 1219 If the root file system corresponding to this menu entry is a ZFS dataset, the 1220 menu entry needs the following option added: 1221 .sp 1222 .in +2 1223 .nf 1224 -B \$ZFS-BOOTFS 1225 .fi 1226 .in -2 1227 .sp 1229 .RE 1231 .sp 1232 .ne 2 1233 .na 1234 \fB\fIboot-args\fR\fR 1235 .ad 1236 .sp .6 1237 .RS 4n 1238 The boot program passes all \fIboot-args\fR to \fBfile\fR. They are not 1239 interpreted by fBbootfR. See fBkernelfR(1M) and fBkmdbfR(1) for 1240 information about the options available with the kernel. 1241 .RE 1243 .sp 1244 .ne 2 1245 .na 1246 \fB\fB/platform/i86pc/kernel/\$ISADIR/unix\fR\fR 1247 .ad 1248 .sp .6

# new/usr/src/man/man1m/boot.1m

19

1249 RS 4n 1250 Name of the kernel to boot. When using the \fBkernel\$\fR token, \fB\$ISADIR\fR 1251 expands to \fBamd64\fR on 64-bit machines, and a null string on other machines. 1252 As a result of this dereferencing, this path expands to the proper kernel for 1253 the machine. 1254 .RE 1256 .SH X86 BOOT SEQUENCE DETAILS 1257 LP 1258 After a PC-compatible machine is turned on, the system firmware in the \fBBIOS 1259 ROM/fR executes a power-on self test (POST), runs \fBBIOS/fR extensions in 1260 peripheral board \fBROMs,\fR and invokes software interrupt INT 19h, Bootstrap. 1261 The INT 19h handler typically performs the standard PC-compatible boot, which 1262 consists of trying to read the first physical sector from the first diskette 1263 drive, or, if that fails, from the first hard disk. The processor then jumps to 1264 the first byte of the sector image in memory. 1265 .SH X86 PRIMARY BOOT 1266 .LP 1267 The first sector on a floppy disk contains the master boot block (GRUB 1268 fBstage1fR. The stage 1 is responsible for loading GRUB fBstage2fR. Now 1269 GRUB is fully functional. It reads and executes the menu file 1270 \fB/boot/grub/menu.lst\fR. A similar sequence occurs for DVD or CD boot, but 1271 the master boot block location and contents are dictated by the El Torito 1272 specification. The El Torito boot also leads to \fBstrap.com\fR, which in turn 1273 loads \fBboot.bin\fR. 1274 .sp 1275 .LP 1267 The first sector on a hard disk contains the master boot block, which contains 1268 the master boot program and the \fBFDISK\fR table, named for the \fBPC\fR 1269 program that maintains it. The master boot finds the active partition in the 1270 \fBFDISK\fR table, loads its first sector (GRUB \fBstage1\fR), and jumps to its 1271 first byte in memory. This completes the standard PC-compatible hard disk boot 1272 sequence. If GRUB fBstagelfR is installed on the master boot block (see the 1273 \fB-m\fR option of \fBinstallgrub\fR(1M)), then \fBstage2\fR is loaded directly 1274 from the Solaris partition regardless of the active partition. 1275 .sp 1276 LP 1277 A similar sequence occurs for DVD or CD boot, but the master boot block location 1278 and contents are dictated by the El Torito specification. The El Torito boot 1279 will then continue in the same way as with the hard disk. 1280 .sp 1281 .LP 1282 Floppy booting is not longer supported. Booting from USB devices follows the 1283 same procedure as with hard disks. 1284 .sp 1285 .T.P 1286 An x86 \fBFDISK\fR partition for the Solaris software begins with a 1287 one-cylinder boot slice, which contains GRUB \fBstage1\fR in the first sector, 1288 the standard Solaris disk label and volume table of contents (VTOC) in the 1289 second and third sectors, and GRUB \fBstage2\fR in the fiftieth and subsequent 1290 sectors. The area from sector 4 to 49 might contain boot blocks for older 1291 versions of Solaris. This makes it possible for multiple Solaris releases on 1292 the same FDISK to coexist. When the \fBFDISK\fR partition for the Solaris 1293 software is the active partition, the master boot program (\fBmboot\fR) reads 1294 the partition boot program in the first sector into memory and jumps to it. It 1295 in turn reads GRUB \fBstage2\fR program into memory and jumps to it. Once the 1296 GRUB menu is displayed, the user can choose to boot an operating system on a 1297 different partition, a different disk, or possibly from the network. 1298 .sp 1299 LP 1300 The behavior is slightly different when a disk is using \fBEFI\fR 1301 partitioning. In that case the GRUB \fBstage1\fR is always installed 1302 in the first sector of the disk, and it always loads \fBstage2\fR from 1303 the partition specified at GRUB installation time. This only works on 1304 partitions containing a ZFS root pool. 1305 .sp

new/usr/src/man/man1m/boot.1m	21 new/usr/src/man/manlm/boot.lm 22
1306 .LP	1372 .in +2
1307 For network booting, the supported method is Intel's Preboot eXecution	1373 .nf
1308 Environment (PXE) standard. When booting from the network using PXE, the syst	tem 1374 \fBboot net\fR
1309 or network adapter BIOS uses DHCP to locate a network bootstrap program	1375 .fi
1310 (\fBpxegrub\fR) on a boot server and reads it using Trivial File Transfer	1376 .in -2
1311 Protocol (TFTP). The BIOS executes the \fBpxegrub\fR by jumping to its first	1377 .sp
1312 byte in memory. The \fBpxegrub\fR program downloads a menu file and presents	
1313 the entries to user.	1379 .sp
1314 .SH X86 KERNEL STARTUP	1380 .LP
1315 .LP	1381 The command below results in no device arguments. The network boot support
1316 The kernel startup process is independent of the kernel loading process. Dur: 1317 kernel startup, console I/O goes to the device specified by the \fBconsole\fI	
1318 property.	1384 .sp
1319 .sp	1385 .in +2
1320 .LP	1386 .nf
1321 When booting from UFS, the root device is specified by the \fBbootpath\fR	1387 \fBboot net:\fR
1322 property, and the root file system type is specified by the \fBfstype\fR	1388 .fi
1323 property. These properties should be setup by the Solaris Install/Upgrade	1389 .in -2
1324 process in $fB/boot/solaris/bootenv.rcfR and can be overridden with the$	1390 .sp
1325 $B-B\$ option, described above (see the $fR(1M)$ man page).	
1326 .sp	1392 .sp
1327 .LP	1393 .LP
1328 When booting from ZFS, the root device is specified by a boot parameter 1329 specified by the $B-B\$ (BSZFS-BOOTFS) fR parameter on either the	1394 The command below results in no device arguments. \fBrarp\fR is the only 1395 network boot support package argument. \fBnetwork-boot-arguments\fR is ignored.
1330 \fBkernel\FR or \fBmodule\fR line in the GRUB menu entry. This value (as with	
1331 all parameters specified by the $\{B-B\}$ (R option) is passed by GRUB to the	1397 .sp
1332 kernel.	1398 .in +2
1333 .sp	1399 .nf
1334 .LP	1400 \fBboot net:rarp\fR
1335 If the console properties are not present, console I/O defaults to \fBscreen	
1336 and $fR$ and $fR$ . The root device defaults to $fR$ and the file 1337 system defaults to $fR$ .	1402 .in -2 1403 .sp
1338 SH EXAMPLES	TID .Sh
1339 .SS "SPARC"	1405 .sp
1340 .LP	1406 .LP
1341 \fBExample 1 \fRTo Boot the Default Kernel In Single-User Interactive Mode	1407 In the command below, the specified device arguments are honored. The network
1342 .sp	1408 boot support package processes arguments in \fBnetwork-boot-arguments\fR.
1343 .LP 1344 To boot the default kernel in single-user interactive mode, respond to the	1410 .sp
1345 \fBok\fR prompt with one of the following:	$1410 \cdot sp$ 1411 .in +2
is is (ibox (in prompt with one of the following)	1412 .nf
1347 .sp	1413 \fBboot net:speed=100,duplex=full\fR
1348 .in +2	1414 .fi
1349 .nf	1415 .in -2
1350 \fBboot\fR \fB\fR\fB-as\fR	1416 .sp
1352 \fBboot\fR \fBdisk3\fR \fB-as\fR	1418 .LP
1353 .fi	1419 \fBExample 3 \fRUsing \fBwanboot\fR with Older PROMs
1354 .in -2	1420 .sp
1355 .sp	1421 .LP
	1422 The command below results in the \fBwanboot\fR binary being loaded from DVD or
1357 LP	1423 CD, at which time \fBwanboot\fR will perform DHCP and then drop into its
1358 $BExample 2 RNetwork Booting with WAN Boot-Capable PROMs 1359 .sp$	1424 command interpreter to allow the user to enter keys and any other necessary 1425 configuration.
1360 .LP	
1361 To illustrate some of the subtle repercussions of various boot command line	1427 .sp
1362 invocations, assume that the $fBnetwork-boot-arguments fR$ are set and that	1428 .in +2
1363 \fBnet\fR is devaliased as shown in the commands below.	1429 .nf
1265	1430 \fBboot cdrom -F wanboot -o dhcp,prompt\fR
1365 .sp	1431 .fi 1432 .in -2
1366 .LP 1367 In the following command, device arguments in the device alias are processed	
1368 the device driver. The network boot support package processes arguments in	21 1.55 · CF
1369 \fBnetwork-boot-arguments\fR.	1435 .SS "x86 (32-bit)"
	1436 .LP
1371 .sp	1437 \fBExample 4 \fRTo Boot the Default Kernel In 32-bit Single-User Interactive

1438 Mode 1439 .sp 1440 LP 1441 To boot the default kernel in single-user interactive mode, edit the GRUB 1442 kernel command line to read: 1444 .sp 1445 .in +2 1446 .nf 1447 kernel /platform/i86pc/kernel/unix -as 1448 .fi 1449 .in -2 1450 .sp 1452 .SS "x86 (64-bit Only)" 1453 .LP 1454 \fBExample 5 \fRTo Boot the Default Kernel In 64-bit Single-User Interactive 1455 Mode 1456 .sp 1457 .LP 1458 To boot the default kernel in single-user interactive mode, edit the GRUB 1459 kernel command line to read: 1461 .sp 1462 .in +2 1463 .nf 1464 kernel /platform/i86pc/kernel/amd64/unix -as 1465 .fi 1466 .in -2 1467 .sp 1469 .LP 1470 \fBExample 6 \fRSwitching Between 32-bit and 64-bit Kernels on 64-bit x86 1471 Platform 1472 .sp 1473 .LP 1474 To be able to boot both 32-bit and 64-bit kernels, add entries for both kernels 1475 to \fB/boot/grub/menu.lst\fR, and use the \fBset-menu\fR subcommand of 1476 fBbootadm fR(1M) to switch. See fBbootadm fR(1M) for an example of the 1477 \fBbootadm set-menu\fR. 1479 .SH FILES 1480 .ne 2 1481 .na 1482 \fB\fB/platform/\fR\fIplatform-name\fR\fB/ufsboot\fR\fR 1483 .ad 1484 .sp .6 1485 .RS 4n 1486 Second-level program to boot from a disk, DVD, or CD 1487 .RE 1489 .sp 1490 .ne 2 1491 .na 1482 \fB\fB/etc/inittab\fR\fR 1483 .ad 1484 .sp .6 1485 .RS 4n 1486 Table in which the \fBinitdefault\fR state is specified 1487 .RE 1489 .sp 1490 .ne 2 1491 .na 1492 \fB\fB/sbin/init\fR\fR 1493 .ad

23

new/usr/src/man/man1m/boot.1m 1494 .sp .6 1495 .RS 4n 1496 Program that brings the system to the fBinitdefaultfR state 1497 .RE 1499 .SS "64-bit SPARC Only" 1500 .ne 2 1501 .na 1502 \fB\fB/platform/\fR\fIplatform-name\fR\fB/kernel/sparcv9/unix\fR\fR 1503 .ad 1504 .sp .6 1505 RS 4n 1506 Default program to boot system. 1507 .RE 1509 .SS "x86 Only" 1510 .ne 2 1511 .na 1512 \fB\fB/boot\fR\fR 1513 .ad 1514 .sp .6 1515 .RS 4n 1516 Directory containing boot-related files. 1517 .RE 1519 .sp 1520 .ne 2 1521 .na 1522 \fB\fB/rpool/boot/grub/menu.lst\fR\fR 1532 \fB\fB/boot/grub/menu.lst\fR\fR 1523 .ad 1524 .sp .6 1525 .RS 4n 1526 Menu of bootable operating systems displayed by GRUB. 1527 .sp 1528 \fBNote:\fR this file is located on the root ZFS pool. While many installs 1529 often name their root zpool 'rpool', this is not required and the 1530 /rpool in the path above should be substituted with the name of 1531 the root pool of your current system. 1532 .RE 1534 .sp 1535 .ne 2 1536 .na 1537 \fB\fB/platform/i86pc/kernel/unix\fR\fR 1538 ad 1539 .sp .6 1540 .RS 4n 1541 32-bit kernel. 1542 .RE 1544 .SS "64-bit x86 Only" 1545 .ne 2 1546 .na 1547 fBfJplatform/i86pc/kernel/amd64/unixfRfR1548 .ad 1549 .sp .6 1550 .RS 4n 1551 64-bit kernel. 1552 .RE 1554 .SH SEE ALSO 1555 .LP  $\label{eq:list} $$1556 \fBkmdb\fR(1), \fBuname\fR(1), \fBbootadm\fR(1M), \fBeeprom\fR(1M), \fBeeprom$ 1557 \fBinit\fR(1M), \fBinstallboot\fR(1M), \fBkernel\fR(1M), \fBmonitor\fR(1M), 1558 \fBshutdown\fR(1M), \fBsvcadm\fR(1M), \fBumountall\fR(1M), \fBzpool\fR(1M),

25

1559  $fBuadmin_{R(2)}, fBbootparams_{R(4)}, fBinittab_{R(4)}, fBvfstab_{R(4)},$ 1560 \fBwanboot.conf\fR(4), \fBfilesystem\fR(5) 1561 .sp 1562 .LP 1563 RFC 903, \fIA Reverse Address Resolution Protocol\fR, 1564 \fBhttp://www.ietf.org/rfc/rfc903.txt\fR 1565 .sp 1566 .LP 1567 RFC 2131, \fIDynamic Host Configuration Protocol\fR, 1568 \fBhttp://www.ietf.org/rfc/rfc2131.txt\fR 1569 .sp 1570 .LP 1571 RFC 2132, \fIDHCP Options and BOOTP Vendor Extensions\fR, 1572 \fBhttp://www.ietf.org/rfc/rfc2132.txt\fR 1573 .sp 1574 .LP 1575 RFC 2396, \fIUniform Resource Identifiers (URI): Generic Syntax\fR, 1576 \fBhttp://www.ietf.org/rfc/rfc2396.txt\fR 1577 .sp 1578 .LP 1579 \fl\fR 1580 .sp 1581 .LP 1582 \fISun Hardware Platform Guide\fR 1583 .sp 1584 .LP 1585 \fIOpenBoot Command Reference Manual\fR 1586 .SH WARNINGS 1587 .LP 1588 The \fBboot\fR utility is unable to determine which files can be used as 1589 bootable programs. If the booting of a file that is not bootable is requested, 1590 the \fBboot\fR utility loads it and branches to it. What happens after that is 1591 unpredictable. 1592 .SH NOTES 1593 .LP 1594 \fIplatform-name\fR can be found using the fB-i/fR option of fBuname fR(1). 1595 \fIhardware-class-name\fR can be found using the \fB-m\fR option of 1596 \fBuname\fR(1). 1597 .sp 1598 .LP 1599 The current release of the Solaris operating system does not support machines 1600 running an UltraSPARC-I CPU.