NAME

bsdinstall - system installer

SYNOPSIS

bsdinstall [*options*] [*target*] [...]

DESCRIPTION

bsdinstall is used for installation of new systems, both for system setup from installation media, e.g., CD-ROMs, and for use on live systems to prepare VM images and jails.

Much like make(1), **bsdinstall** takes a target and possible parameters of the target as arguments. If invoked with no arguments, it will invoke the **auto** target, which provides a standard interactive installation, invoking the others in sequence. To perform a scripted installation, these subtargets can be invoked separately by an installation script.

OPTIONS

bsdinstall supports the following options, global to all targets:

-D *file* Provide a path for the installation log file (overrides BSDINSTALL_LOG). See *ENVIRONMENT VARIABLES* for more information on BSDINSTALL_LOG.

TARGETS

Most of the following targets are only useful for scripting the installer. For interactive use, most users will be interested only in the **auto**, **jail**, and **script** targets.

auto Run the standard interactive installation, including disk partitioning.

jail *destination* Sets up a new chroot system at *destination*, suitable for use with jail(8). Behavior is generally similar to **auto**, except that disk partitioning and network setup are skipped and a kernel is not installed into the new system.

script *script* Runs the installation script at *script*. See *SCRIPTING* for more information on this target.

keymap If the current controlling TTY is a syscons(4) or vt(4) console, asks the user to set the current keymap, and saves the result to the new system's *rc.conf*.

hostname Prompts the user for a host name for the new system and saves the result to the new system's *rc.conf*. If BSDINSTALL_CONFIGCURRENT is set, also sets the host name of the current system.

netconfig Interactively configures network interfaces (first invoking **wlanconfig** on wireless interfaces), saving the result to the new system's *rc.conf* and *resolv.conf*. If BSDINSTALL_CONFIGCURRENT is set, also configures the network interfaces of the current system to match.

autopart Provides the installer's interactive guided disk partitioner for single-disk installations. Defaults to UFS.

bootconfig Detects an appropriate partition and installs UEFI boot loader files.

zfsboot Provides a ZFS-only automatic interactive disk partitioner. Creates a single **zpool** with separate datasets for */home*, */tmp*, */usr*, */usr/ports*, */usr/src*, and */var*. Optionally can set up geli(8) to encrypt the disk.

partedit Provides the installer's interactive manual disk partitioner with an interface identical to sade(8). Supports multiple disks as well as UFS, ZFS, and FAT file systems. ZFS is set up with one pool and dataset per partition.

scriptedpart *parameters*

Sets up disks like **autopart** and **partedit**, but non-interactively according to the disk setup specified in *parameters*. Each disk setup is specified by a three-part argument:

disk [*scheme*] [*{partitions}*]

Multiple disk setups are separated by semicolons. The *disk* argument specifies the disk on which to operate (which will be erased), or the special value *DEFAULT*, which will result in either a selection window (as in **autopart**) for the destination disk or, if there is only one possible disk, will automatically select it. The *scheme* argument specifies the gpart(8) partition scheme to apply to the disk. If *scheme* is unspecified, **scriptedpart** will apply the default bootable scheme on your platform. The *partitions* argument is also optional and specifies how to partition *disk*. It consists of a comma-separated list of partitions to create enclosed in curly braces. Each partition declaration takes the form

size type [*mount point*]

size specifies the partition size to create in bytes (K, M, and G suffixes can be appended to specify kilobytes, megabytes, and gigabytes respectively), while the *auto* keyword causes the partition to take all the remaining space on the disk. The *type* option chooses the gpart(8) filesystem type, e.g., freebsd-ufs, freebsd-zfs, or freebsd-

swap. The optional *mount point* argument sets where the created partition is to be mounted in the installed system. As an example, a typical invocation looks like:

bsdinstall scriptedpart ada0 { 20G freebsd-ufs /, 4G freebsd-swap, 20G freebsd-ufs /var, auto freebsd-ufs /usr }

Note that the list of partitions should *not* include boot partitions (e.g. EFI system partitions), which will be created automatically on whatever disk includes /.

A shorter invocation to use the default partitioning (as **autopart** would have used) on the same disk:

bsdinstall scriptedpart ada0

or, even shorter:

bsdinstall scriptedpart DEFAULT

mount Mounts the file systems previously configured by **autopart**, **partedit**, or **scriptedpart** under BSDINSTALL_CHROOT.

distfetch Fetches the distributions in DISTRIBUTIONS to BSDINSTALL_DISTDIR from BSDINSTALL_DISTSITE.

checksum Verifies the checksums of the distributions listed in DISTRIBUTIONS against the distribution manifest.

distextract Extracts the distributions listed in DISTRIBUTIONS into BSDINSTALL_CHROOT.

rootpass Interactively invokes passwd(1) in the new system to set the root user's password.

adduser Interactively invokes adduser(8) in the new system.

time Interactively sets the time, date, and time zone of the new system.

services Queries the user for the system daemons to begin at system startup, writing the result into the new system's *rc.conf*.

entropy Reads a small amount of data from */dev/random* and stores it in a file in the new system's root directory.

config Installs the configuration files destined for the new system, e.g., rc.conf(5) fragments generated by **netconfig**, etc.) onto the new system.

ENVIRONMENT VARIABLES

The following environment variables control various aspects of the installation process. Many are used internally during installation and have reasonable default values for most installation scenarios. Others are set by various interactive user prompts, and can be usefully overridden when making scripted or customized installers.

TMPDIR The directory to use for temporary files. Default: "*/tmp*"

DISTRIBUTIONS The set of distributions to install, e.g., "base.txz kernel.txz ports.txz". Default: unset

PARTITIONS The partitioning of the disk onto which the system is being installed. See **scriptedpart** of the *TARGETS* section for format details. If this variable is unset, the installer will use the default partitioning as in **autopart**. Default: unset

BSDINSTALL_DISTDIR The directory in which the distribution files can be found (or to which they should be downloaded). Default: "*/usr/freebsd-dist*"

BSDINSTALL_DISTSITE URL from which the distribution files should be downloaded if they are not already present in the directory defined by BSDINSTALL_DISTDIR. This should be a full path to the files, including architecture and release names. Most targets, e.g., **auto** and **jail**, that prompt for a FreeBSD mirror will skip that step if this variable is already defined in the environment. Example: *https://download.freebsd.org/ftp/releases/powerpc/powerpc64/13.1-RELEASE/* or *http://ftp-archive.freebsd.org/pub/FreeBSD-Archive/old-releases/amd64/12.2-RELEASE/*.

BSDINSTALL_CHROOT The directory into which the distribution files should be unpacked and the directory at which the root file system of the new system should be mounted. Default: "*/mnt*"

BSDINSTALL_LOG Path to a log file for the installation. Default: "*$TMPDIR/bsdinstall_log*"

BSDINSTALL_TMPETC Directory where files destined for the new system's */etc* will be stored until the **config** target is executed. If this directory does not already exist, it will be created. Default: "*$TMPDIR/bsdinstall_etc*"

BSDINSTALL_TMPBOOT
: Directory where files destined for the new system's */boot* will be stored until the **config** target is executed. If this directory does not already exist, it will be created. Default: "*$TMPDIR/bsdinstall_boot*"

ROOTPASS_ENC
: Encrypted string to set the root password to in the format expected by pw(8) **-H** *0*. This option is used if both it and ROOTPASS_PLAIN are set.

ROOTPASS_PLAIN
: Plain text string to set the root password to.

ZFSBOOT_POOL_NAME
: Name for the pool containing the base system. Default: "zroot"

ZFSBOOT_POOL_CREATE_OPTIONS
: Options to be used when creating the base system's pool. Each option must be preceded by the -O flag to be taken into consideration or the pool will not be created due to errors using the command **zpool**. Default: "-O compress=lz4 -O atime=off"

ZFSBOOT_BEROOT_NAME
: Name for the boot environment parent dataset. This is a non-mountable dataset meant to be a parent dataset where different boot environment are going to be created. Default: "ROOT"

ZFSBOOT_BOOTFS_NAME
: Name for the primary boot environment, which will be the default boot environment for the system. Default: "default"

ZFSBOOT_VDEV_TYPE
: The type of pool to be created for the base system. This variable can take one of this values: stripe (No redundancy), mirror (n-Way mirroring), raid10 (RAID 1+0 - n x 2-Way Mirrors), raidz1 (RAID-Z1 - Single Redundancy RAID), raidz2 (RAID-Z2 - Double Redundancy RAID) or raidz3 (RAID-Z3 Triple Redundancy RAID). Default: "stripe"

ZFSBOOT_FORCE_4K_SECTORS
: Indicates either the pool will use 4K or 512 sectors. If this variable is not empty, 4K sectors will be used. Default: "1"

ZFSBOOT_GELI_ENCRYPTION
: If this variable is not empty, it will use geli(8) to encrypt the root pool,

enabling automatically the ZFSBOOT_BOOT_POOL variable. Default: ""

ZFSBOOT_GELI_KEY_FILE
: Path to the geli(8) keyfile used to encrypt the pool where the base system is stored. Default: "*/boot/encryption.key*"

ZFSBOOT_BOOT_POOL
: If set, a separated boot pool will be created for the kernel of the system and loader(8). Default: unset

ZFSBOOT_BOOT_POOL_CREATE_OPTIONS
: Options to use when creating the boot pool, when enabled (See ZFSBOOT_BOOT_POOL). Default: unset

ZFSBOOT_BOOT_POOL_NAME
: Name for the optional boot pool when it is enabled, (See ZFSBOOT_BOOT_POOL). Default: "bootpool"

ZFSBOOT_BOOT_POOL_SIZE
: Size of the boot pool when it is enabled (See ZFSBOOT_BOOT_POOL). Default: "2g"

ZFSBOOT_DISKS
: Disks to be used for the base system, including the boot pool. This variable must only be used on a scripted installation. See *SCRIPTING* for more information. Default: unset

ZFSBOOT_SWAP_SIZE
: Size of the swap partition on each block device. This variable will be passed to gpart(8); which supports SI unit suffixes. Default: "2g"

ZFSBOOT_SWAP_ENCRYPTION
: If set, enables the encryption of the swap partition using geli(8). Default: ""

ZFSBOOT_SWAP_MIRROR
: If set, enables a swap mirroring using gmirror(8). Default: unset

ZFSBOOT_DATASETS
: ZFS datasets to be created on the root zpool, it requires the following datasets: */tmp*, */var/tmp*, */$ZFSBOOT_BEROOT_NAME/$ZFSBOOT_BOOTFS_NAME*. See *ZFS DATASETS* for more information about how to populate this variable and its default value.

ZFSBOOT_CONFIRM_LAYOUT
: If set and the installation is interactive, allow the user to confirm the layout before continuing with the installation. Default: "1"

SCRIPTING

bsdinstall supports unattended, or minimally-attended, installations using scripting. This can be used with either modified physical installation media or with diskless(8) installations over the network; information on preparing such media can be found in *BUILDING AUTOMATIC INSTALL MEDIA*

Scripted installations follow an essentially identical path to interactive installations, though with some minor feature differences (for example, scripted installations do not support fetching of remote distribution files since scripted installations normally install the same files and the distributions can be added directly to the installation media). **bsdinstall** scripts consist of two parts: a *preamble* and a *setup script*. The preamble sets up the options for the installation (how to partition the disk[s], which distributions to install, etc.) and the optional second part is a shell script run under chroot(8) in the newly installed system before **bsdinstall** exits. The two parts are separated by the usual script header (#!), which also sets the interpreter for the setup script.

A typical bsdinstall script, using the default filesystem layout and the UFS filesystem, looks like this:

```
PARTITIONS=DEFAULT
DISTRIBUTIONS="kernel.txz base.txz"

#!/bin/sh
sysrc ifconfig_DEFAULT=DHCP
sysrc sshd_enable=YES
pkg install puppet
```

For a scripted installation involving a ZFS pool spanning multiple disks, the script instead looks like this:

```
DISTRIBUTIONS="kernel.txz base.txz"
export ZFSBOOT_VDEV_TYPE=stripe
export ZFSBOOT_DISKS="ada0 ada1"
export nonInteractive="YES"

#!/bin/sh
echo "ifconfig_DEFAULT=DHCP" >> /etc/rc.conf
echo "sshd_enable=YES" >> /etc/rc.conf
pkg install puppet
```

On FreeBSD release media, such a script placed at */etc/installerconfig* will be run at boot time and the system will be rebooted automatically after the installation has completed. This can be used for unattended network installation of new systems; see diskless(8) for details.

PREAMBLE

The preamble consists of installer settings. These control global installation parameters (see *ENVIRONMENT VARIABLES*) as well as disk partitioning. The preamble is interpreted as a sh(1) script run at the very beginning of the install. If more complicated behavior than setting these variables is desired, arbitrary commands can be run here to extend the installer. In addition to the variables in *ENVIRONMENT VARIABLES*, in particular DISTRIBUTIONS, the preamble can contain a variable PARTITIONS which is passed to the **scriptedpart** target to control disk setup.

Alternatively, to use **zfsboot** instead of **partedit**, the preamble can contain the variable ZFSBOOT_DATASETS instead of PARTITIONS (see below). If using **zfsboot**, the variables ZFSBOOT_DISKS and ZFSBOOT_VDEV_TYPE must be set to create the pool of disks for the base system. Usually, for a mirrored booting disk, these two variables look like this:

```
ZFSBOOT_DISKS="ada0 ada1"
ZFSBOOT_VDEV_TYPE=mirror
```

Remember to export all the variables for the **zfsboot** command, otherwise installation will fail.

SETUP SCRIPT

Following the preamble is an optional shell script, beginning with a #! declaration. This script will be run at the end of the installation process inside a chroot(8) environment in the newly installed system and can be used to set up configuration files, install packages, etc. Note that newly configured system services, e.g., networking have not been started in the installed system at this time and only installation host services are available.

ZFS DATASETS

If using **zfsboot** in an installation script, the **zfsboot** partitioning tool takes the ZFSBOOT_DATASETS variable to create the ZFS datasets on the base system. This variable definition can become large if the pool contains many datasets. The default value of ZFSBOOT_DATASETS is:

```
# DATASET         OPTIONS (comma or space separated; or both)

# Boot Environment [BE] root and default boot dataset
/$ZFSBOOT_BEROOT_NAME                                   mountpoint=none
/$ZFSBOOT_BEROOT_NAME/$ZFSBOOT_BOOTFS_NAME      mountpoint=/
```

```
# Home directories separated so they are common to all BEs
/home                   mountpoint=/home

# Compress /tmp, allow exec but not setuid
/tmp                    mountpoint=/tmp,exec=on,setuid=off

# Do not mount /usr so that 'base' files go to the BEROOT
/usr                    mountpoint=/usr,canmount=off

# Ports tree
/usr/ports setuid=off

# Source tree (compressed)
/usr/src

# Create /var and friends
/var                    mountpoint=/var,canmount=off
/var/audit exec=off,setuid=off
/var/crash exec=off,setuid=off
/var/log   exec=off,setuid=off
/var/mail  atime=on
/var/tmp   setuid=off
```

The first column is the name of the dataset to be created as part of the ZFSBOOT_POOL_NAME pool and the remainder of each line contains the options to be set on each dataset. If multiple options are given, they can be separated by either commas or whitespace; everything following a pound/hash character is ignored as a comment.

BUILDING AUTOMATIC INSTALL MEDIA

If building automatic install media, use tar to extract a release ISO:

```
mkdir release-media
tar -C release-media -xvf FreeBSD-13.0-RELEASE-amd64-disc1.iso
```

Then place a script as above in *etc/installerconfig*

This directory can then be used directly as an NFS root for diskless(8) installations or it can be rebuilt into an ISO image using the release scripts in */usr/src/release*. For example, on amd64:

```
sh /usr/src/release/amd64/mkisoimages.sh -b '13_0_RELEASE_AMD64_CD' output.iso
release-media
```

HISTORY

This version of **bsdinstall** first appeared in FreeBSD 9.0.

AUTHORS

Nathan Whitehorn <*nwhitehorn@FreeBSD.org*>
Devin Teske <*dteske@FreeBSD.org*>
Allan Jude <*allanjude@FreeBSD.org*>