

4. ZFS (ZFS Datasets)

1. 在 /etc/fstab 文件中添加以下配置，以启用 NTFS 文件系统支持：


```

# 在 /etc/fstab 文件中添加以下配置
ntfs-3g /mnt/ntfs /dev/sdb1 defaults 0 0

```

[illegible]

```











# 检查 /var 分区是否使用 UFS 文件系统
# 如果 /var 分区使用 UFS 文件系统，则使用 tuneufs(8) 命令
# 对 /var 分区进行碎片整理
# 如果 /var 分区使用 XFS 文件系统，则使用 xfs_fsr(8) 命令
# 对 /var 分区进行碎片整理
# 如果 /var 分区使用 ZFS 文件系统，则使用 zfs sendrecv(8) 命令
# 对 /var 分区进行碎片整理




```


 . ZFS

 .

????? (Datasets)

ZFS 的 数据 块 大小 是 128K， 而 元数据 块 大小 是 1K。 这 意味着 元数据 的 写入 速度 比 数据 的 写入 速度 快 得多。 因此， 在 写入 数据 时， 元数据 的 写入 速度 是 瓶颈。 这 就是 为什么 在 写入 数据 时， 元数据 的 写入 速度 是 瓶颈。 这 就是 为什么 在 写入 数据 时， 元数据 的 写入 速度 是 瓶颈。

zfs(8)

????? ?? (Dataset Types)

而 ZFS 的 文件系统 (filesystems), 卷 (volumes), 快照 (snapshots), 克隆 (clones), 书签 (bookmarks) 这 5 个 概念 都是 相互 关联 的 。

而 我们 通常 使用 的 是 卷 (volumes) 和 快照 (snapshots) 。 ZFS 文件系统 的 一个 重要 特性 是 它 支持 权限 控制 , setuid 权限 的 设置 和 文件 的 加密 功能 。 此外 , ZFS 还支持 快照 的 创建 和 删除 , 以及 文件 的 克隆 和 删除 。 NFSv4 的 支持 也是 一个 重要 的 特性 , chflags(2) 的 支持 也是 一个 重要 的 特性 。

ZFS 的 一个 重要 特性 是 **zvol** 的 支持 。 它 支持 通过 iSCSI 的 方式 来 使用 ZFS 的 卷 (volumes) 。 ZFS 的 一个 重要 特性 是 它 支持 快照 (snapshots) 的 创建 和 删除 。 Zvol 的 支持 也是 一个 重要 的 特性 , FreeBSD 的 支持 也是 一个 重要 的 特性 。

此外 , ZFS 还支持 文件 的 加密 和 解密 , 以及 文件 的 压缩 和 解压 。 此外 , ZFS 还支持 文件 的 删除 和 恢复 。 此外 , ZFS 还支持 文件 的 克隆 和 删除 。 此外 , ZFS 还支持 文件 的 书签 (bookmarks) 的 创建 和 删除 。

此外 , ZFS 还支持 文件 的 权限 控制 , 以及 文件 的 加密 和 解密 。 此外 , ZFS 还支持 文件 的 压缩 和 解压 。 此外 , ZFS 还支持 文件 的 删除 和 恢复 。 此外 , ZFS 还支持 文件 的 克隆 和 删除 。 此外 , ZFS 还支持 文件 的 书签 (bookmarks) 的 创建 和 删除 。

??? ??? ??? ??? ?????? (Why Do I Want Datasets?)

我们 通常 使用 的 是 卷 (volumes) 和 快照 (snapshots) 。 ZFS 的 一个 重要 特性 是 它 支持 权限 控制 , setuid 权限 的 设置 和 文件 的 加密 功能 。 此外 , ZFS 还支持 快照 的 创建 和 删除 , 以及 文件 的 克隆 和 删除 。 NFSv4 的 支持 也是 一个 重要 的 特性 , chflags(2) 的 支持 也是 一个 重要 的 特性 。

而 ZFS 的 一个 重要 特性 是 **zvol** 的 支持 。 它 支持 通过 iSCSI 的 方式 来 使用 ZFS 的 卷 (volumes) 。 ZFS 的 一个 重要 特性 是 它 支持 快照 (snapshots) 的 创建 和 删除 。 Zvol 的 支持 也是 一个 重要 的 特性 , FreeBSD 的 支持 也是 一个 重要 的 特性 。

此外 , ZFS 还支持 文件 的 加密 和 解密 , 以及 文件 的 压缩 和 解压 。 此外 , ZFS 还支持 文件 的 删除 和 恢复 。 此外 , ZFS 还支持 文件 的 克隆 和 删除 。 此外 , ZFS 还支持 文件 的 书签 (bookmarks) 的 创建 和 删除 。

此外 , ZFS 还支持 文件 的 权限 控制 , 以及 文件 的 加密 和 解密 。 此外 , ZFS 还支持 文件 的 压缩 和 解压 。 此外 , ZFS 还支持 文件 的 删除 和 恢复 。 此外 , ZFS 还支持 文件 的 克隆 和 删除 。 此外 , ZFS 还支持 文件 的 书签 (bookmarks) 的 创建 和 删除 。

此外 , ZFS 还支持 文件 的 加密 和 解密 , 以及 文件 的 压缩 和 解压 。 此外 , ZFS 还支持 文件 的 删除 和 恢复 。 此外 , ZFS 还支持 文件 的 克隆 和 删除 。 此外 , ZFS 还支持 文件 的 书签 (bookmarks) 的 创建 和 删除 。

如何 实现 ?

我们 在 这个 项目 中 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 . 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

如何 实现 (Viewing Datasets)

zfs list 我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .

```
$ zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
mypool                             420M  17.9G   96K    none
mypool/R00T                         418M  17.9G   96K    none
mypool/R00T/default                 418M  17.9G  418M    /
...
```

我们 使用 了 很多 不同的 技术 , 但 我们 在 这个 项目 中 使用 了 很多 不同的 技术 .


```
$ zfs create mypool/lamb/baby
```

이제 우리는 'lamb' / 'baby' 데이터셋을 생성했습니다. , 데이터셋은 'lamb' 데이터셋의 하위 데이터셋입니다.

?? ??? (Creating Volumes)

-V 옵션은 데이터셋의 크기를 지정합니다. **zfs create** 명령을 사용하여 데이터셋을 생성합니다. 이 데이터셋은 4GB의 크기를 가집니다.

```
$ zfs create -V 4G mypool/avolume
```

Zvols은 ZFS 데이터셋을 사용하여 데이터를 저장하는 데 사용됩니다. **-t volume** 옵션을 사용하여 ZFS 데이터셋을 Zvol로 생성합니다. **zfs list** 명령을 사용하여 Zvol을 확인합니다.

```
$ zfs list mypool/avolume
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
mypool/avolume	4.13G	17.9G	64K	-

Zvol은 ZFS 데이터셋을 사용하여 데이터를 저장하는 데 사용됩니다. 이 Zvol은 4GB의 크기를 가집니다. Zvol은 4.13GB의 크기를 가집니다.

Zvol은 ZFS 데이터셋을 사용하여 데이터를 저장하는 데 사용됩니다. 이 Zvol은 /dev/zvol/0의 경로를 가집니다. 이 Zvol은 4GB의 크기를 가집니다.

```
$ ls -al /dev/zvol/mypool/avolume
```

```
crw-r----- 1 root operator 0x4d Mar 27 20:22 /dev/zvol/mypool/avolume
```

이제 우리는 **newfs(8)** 명령을 사용하여 Zvol에 새로운 파일 시스템을 생성합니다. 이 파일 시스템은 Zvol에 생성됩니다.

????? ?? ?? (Renaming Datasets)

zfs rename 명령을 사용하여 데이터셋의 이름을 변경합니다. 이 명령은 데이터셋의 이름을 변경합니다.

```
$ zfs rename db/production db/old
```

```
$ zfs rename db/testing db/production
```

이제 우리는 **-f** 옵션을 사용하여 데이터셋의 이름을 변경합니다. 이 옵션은 데이터셋의 이름을 변경합니다. 이 옵션은 데이터셋의 이름을 변경합니다.

이제 우리는 데이터를 저장할 수 있습니다.

????? ???? (Moving Datasets)

00000000 ZFS 0000 0000 00 0000 0000 00000000 0 000 00 0000 00 0
 0000 . 000 00000 000 00000 000000 00 000 000 0 0000 .
 00000000 000 000 000 0000 0000 .

```
$ zfs rename zroot/var/db/mysql zroot/important/mysql
```

[illegible]




















????? ???? (Destroying Datasets)

 ?

zfs destroy

 .

```
$ zfs destroy db/old
```

[illegible]

00000000 0000 0 00 00 00000 000 00000 -v 0 -n 0000 000 0 0000 . -v
 0000 0000 000 00 000 000 0000 , -n 0 zfs(8) 0000 00000 00000 .
 0 0 0000 0000 0000 00 0 000 000 000 00000 00000 .

ZFS ?? (ZFS Properties)

[illegible][illegible]

?? ?? (Viewing Properties)

```
zfs(8)  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 10
```

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb	compression	lz4	inherited from mypool

SOURCE 00 0 000000 . 00 000 0 000 ZFS 000000 0000 000 000000 . 00
000 0000 0 000000 0 000 00000 00000 00000 . 00 000 000000
0000 0 000000 , 000000 000 0000 0 000 0000 000 000000 . 000
000 0 00 0000 00 "00 /00 00 "00 0000 00 00 0000000 00000 .

The diagram illustrates the output of the `zfs get` command. It shows two rows of property names and their corresponding values, represented by boxes. The first row contains: `mountpoint`, `quota`, `refreservation`, `smbsharename`, `zfs.get`, `usedbydataset`, `usedbytes`, `usedspace`, `usedspace_limit`, `usedspace_reservation`, and `usedspace_warnfree`. The second row contains: `mountpoint`, `quota`, `refreservation`, `smbsharename`, `zfs.get`, `usedbydataset`, `usedbytes`, `usedspace`, `usedspace_limit`, `usedspace_reservation`, and `usedspace_warnfree`.

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb	type	filesystem	-
mypool/lamb	creation	Fri Mar 27 20:05 2015	-
mypool/lamb	used	192K	-
...			

```
$ zfs get quota,reservation zroot/home
```

NAME	PROPERTY	VALUE	SOURCE
zroot/home	quota	none	local
zroot/home	reservation	none	default

NAME	QUOTA	RESERV
db	none	none
zroot	none	none
zroot/R00T	none	none
zroot/R00T/default	none	none

```
...
zroot/var/log          100G      20G
...
```

zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。

?? ?? (Changing Properties)

zfs set 命令 用于 设置 zfs 实例 的属性。格式 如下：
compression 属性 设置为 **off**。

```
$ zfs set compression=off mypool/lamb/baby
```

zfs get 命令 用于 获取 zfs 实例 的属性。格式 如下：

```
$ zfs get compression mypool/lamb/baby
```

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb/baby	compression	off	local

zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。

?? ?? ?? (Read-Only Properties)

ZFS 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。ZFS 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。ZFS 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。

?? ??? ? (Filesystem Properties)

zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。

atime

zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。zfs 的 属性 分为 本地属性 和 全局属性 两种。本地属性 是指 只针对 某个 zfs 实例 生效 的属性。全局属性 是指 针对 整个 zfs 实例 生效 的属性。

我们使用 `com.allanjude` 命名空间，并设置 `backup_ignore` 属性。

我们使用 `zfs` 命令来设置属性。

```
$ zfs set com.allanjude:backup_ignore=on mypool/lamb
```

我们使用 `zfs` 命令来验证属性。

Parent/Child Relationships

我们使用 `zfs` 命令来设置属性。

```
$ zfs get -r compression mypool/lamb
```

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb	compression	lz4	inherited from mypool
mypool/lamb/baby	compression	off	local

我们使用 `zfs` 命令来设置属性。

zfs inherit 命令用于继承属性。

```
$ zfs inherit compression mypool/lamb/baby
```

```
$ zfs get -r compression mypool/lamb
```

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb	compression	lz4	inherited from mypool
mypool/lamb/baby	compression	lz4	inherited from mypool

我们使用 `zfs` 命令来设置属性。

我们使用 `zfs` 命令来设置属性。

```
$ zfs set compression=gzip-9 mypool/lamb
```

```
$ zfs get -r compression mypool/lamb
```

NAME	PROPERTY	VALUE	SOURCE
mypool/lamb	compression	gzip-9	local
mypool/lamb/baby	compression	gzip-9	inherited from mypool/lamb

1. 我们使用 `gzip-9` 属性来压缩文件。

2. 继承和重命名 (Inheritance and Renaming)

我们使用 `zfs create` 命令来创建新的文件系统。

我们使用 `zfs get` 命令来查看文件系统的属性。

```

$ zfs create mypool/second
$ zfs get compress mypool/second
NAME          PROPERTY      VALUE  SOURCE
mypool/second  compression   lz4    inherited from mypool
```

我们使用 `zfs rename` 命令来重命名文件系统。

```

$ zfs rename mypool/lamb/baby mypool/second/baby
$ zfs get -r compression mypool/second
NAME          PROPERTY      VALUE  SOURCE
mypool/second  compression   lz4    inherited from mypool
mypool/second/baby  compression   lz4    inherited from mypool
```

我们使用 `zfs inherit` 命令来继承属性。

我们使用 `zfs set` 命令来设置属性。

3. 移除属性 (Removing Properties)

我们使用 `zfs unset` 命令来移除属性。

我们使用 `zfs inherit` 命令来继承属性。

```

$ zfs inherit com.allanjude:backup_ignore mypool/lamb
```

我们使用 `zfs get` 命令来查看文件系统的属性。

zfs inherit 命令 用于 设置 文件系统 的 属性 继承 策略。 默认 情况下， 子 文件系统 会 继承 父 文件系统的 属性。

```
$ zfs inherit -r compression mypool
```

zfs inherit 命令 还可以 用于 设置 文件系统 的 其他 属性， 例如 压缩 策略。

ZFS 挂载与卸载 (Mounting ZFS Filesystems)

在 FreeBSD 中， ZFS 文件系统的 挂载 和 卸载 操作 是通过 使用 `mount` 和 `umount` 命令 来完成的。 挂载 操作 通常 是在 系统 启动 时 通过 `/etc/fstab` 文件 来配置 的。 对于 CD-ROM 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

对于 ZFS 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

```
$ zfs get mountpoint zroot/usr/home
```

NAME	PROPERTY	VALUE	SOURCE
zroot/usr/home	mountpoint	/usr/home	inherited from zroot/usr

zfs get 命令 用于 获取 文件系统的 属性 值。 例如， 使用 `zfs get mountpoint zroot/usr/home` 可以 获取 `zroot/usr/home` 文件系统的 挂载 点。

在 FreeBSD 中， ZFS 文件系统的 挂载 和 卸载 操作 是通过 使用 `mount` 和 `umount` 命令 来完成的。 挂载 操作 通常 是在 系统 启动 时 通过 `/etc/fstab` 文件 来配置 的。 对于 CD-ROM 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

对于 ZFS 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

对于 ZFS 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

对于 ZFS 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

对于 ZFS 文件系统， 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。 挂载 操作 通常 是在 用户 手动 插入 光盘 后 进行的。

canmount off 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

如果 数据 集 被 限制 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

canmount 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

??? ???? ?? ????? (Datasets without Mount Points)

ZFS 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

FreeBSD 10.1 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

```
$ zfs mount
zroot/R00T/default /
zroot/tmp /tmp
zroot/usr/home /usr/home
zroot/usr/ports /usr/ports z
root/usr/src /usr/src
...
```

/usr 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

zfs list 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

```
$ zfs list -o name,canmount,mountpoint -r zroot/usr
NAME          CANMOUNT  MOUNTPOINT
zroot/usr      off       /usr
zroot/usr/home on        /usr/home
zroot/usr/ports on        /usr/ports
zroot/usr/src  on        /usr/src
```

canmount off 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

如果 数据 集 被 限制 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

??? ??? ???? ?? ?? ????? (Multiple Datasets with the Same Mount Point)

canmount off 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

canmount off 表示 数据 集 可以 被 挂载 到 任何 位置 。 如果 数据 集 被 挂载 到 指定 的 位置 ， 那么 它 就 是 被 限制 的 。

[illegible]

```
FreeBSD  [ ] [ ] [ ] [ ] [ ] [ ] zroot [ ] [ ] [ ] [ ] . [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

[illegible]

0 00 00 0000 000 00 /opt 0000 000 00 0000 . 00 000 0 00
 0000 0000 0000 00 00 00 00 00 0 . 00 00000 0000 00
 0000 . 00 000 0000 0000 0000 000 000 000 .

```
$ zfs create db/programs # zfs create db/data
```

□□ □ □ □□□□ □□ /opt□ □□ □□ □□ □□ □ □ □□□□ .

```
$ zfs set canmount=off db/programs
$ zfs set mountpoint=/opt db/programs
```

```
$ zfs set readonly=on db/programs
```

[illegible]

```
$ zfs set canmount=off db/data
$ zfs set mountpoint=/opt db/data
$ zfs set setuid=off db/data
$ zfs set exec=off db/data
```

`.db/programs`

```
$ zfs create db/programs/bin
$ zfs create db/programs/sbin
$ zfs create db/data/test
$ zfs create db/data/production
```

□□ /opt□ □□□□ 2□□ □□□□ 2□ , □ 4□□ □□□□□ □□□□ □□□□ . □□□□
 □□□□ □ □□□□□ □□□□ □□□□□□□□ . □□□ □□ □□□ □□□ □□ □□ □ □□□□ □ □□
 □□ □□□ □ □□ □□□□□ . □□□□ □□ □□□□ □□□□ □□□□ □□□ □□ □□□□□ □□ □□

 mount(8) :

```
$ mount -t zfs mypool/second /tmp/second
```

```

# /etc/fstab ZFS
#
# .  .  .
# .  zfs  .  .  .  noatime, noexec, readonly  ro,
# nosuid  .  .  .  .  (  atime, exec, rw, suid  .  .  .  .
# ZFS  .  .  .  fsck  .  .  .  /tmp  .
# scratch/junk nosuid  .  .  .  /etc/fstab  .

```

```
scratch/junk /tmp nosuid 2 0
```

1. 在 `/etc/fstab` 文件中添加以下配置：

ZFS ?? ?? (Tweaking ZFS Volumes)

Zvolí-li se n z N a m z M , pak n a m tvoří dvojici z $N \times M$.
 Každá dvojice z $N \times M$ je tvořena právě jedním n z N a jedním m z M .

?? ?? (Space Reservations)

[illegible][illegible][illegible]

```
zfs create -V [size] -s [compression] [dataset] [permissions] . [quota] [inheritance] [mountpoint]
```

Zvol ?? (Zvol Mode)

```
FreeBSD| | | | | | | | | | zvol| geom(4) | | | | | | | | | .  
volmode | | | | | | | | | | .
```


通过 **volmode** 指定 `dev` 设备名称，如 `/dev/sda`。通过 `geom` 指定 GEOM 设备名称，如 `geom`。通过 `dev` 指定设备名称，如 `dev`。通过 `none` 指定无设备。

volmode 指定 `none` 表示无设备。ZFS 设备名称，如 `zvol`，表示无设备。

volmode 指定 `default` 表示默认设备。通过 `sysctl vfs.zfs.vol.mode` 指定设备名称。通过 `zvol` 指定设备名称，如 `1`，指定 `geom`，`2` 指定 `dev`，`3` 指定 `none`。

通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

Dataset Integrity

ZFS 通过 `VDEV` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

Checksums

ZFS 通过 `on` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

通过 `on` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

通过 `fletcher4` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

`off` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

`noparity` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

通过 `ZFS` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。通过 `zfs rename` 指定设备名称，如 `zvol`。

```
$ zfs list mypool/lamb
```

NAME	USED	AVAIL	REFER	MOUNTPOINT
------	------	-------	-------	------------

```
mypool/lamb 30.2M 13.7G 30.1M /lamb
```











□ □ □ □ 30MB □ , □ □ □ □ □ □ 10, □ □ 10MB □ □ 2□ 20□ □ .

ls(1) □ □ □ □ □ □ :

```
$ ls -l /lamb/random*
-rw-r--r-- 1 root wheel 10485760 Apr 6 15:27 /lamb/random1
-rw-r--r-- 1 root wheel 10485760 Apr 6 15:29 /lamb/random2
```








????? ??? (Metadata Redundancy)

이러한 구성을 사용하면, VDEV 구성을 RAID-Z로 구성하여, 데이터의 중복성을 높일 수 있습니다. 이 구성은 3개의 VDEV를 사용하여, RAID-Z 구성을 구성합니다. 이 구성은 3개의 VDEV를 사용하여, RAID-Z 구성을 구성합니다. 이 구성은 3개의 VDEV를 사용하여, RAID-Z 구성을 구성합니다.

redundant_metadata           .

redundant_metadata all() ZFS .
 .

redundant_metadata most ZFS

redundant_metadata *most*  **copies** 3        ,
ZFS    6      4  .

□ □□ □□□□□ □ □□□□□ □ □□□□□ □ □ □□ □□ □□□□□ .
 □□□ □ □□ □□ □□□□ □□ □□ □□□□□ □□ □□ □□□ □□
 □□□□□ □□ □ □□ □□ □□ □□ □ □□□ . □ □ □□ □□ □□ □□ □□ □□ □□
 □□□ □□□□ .