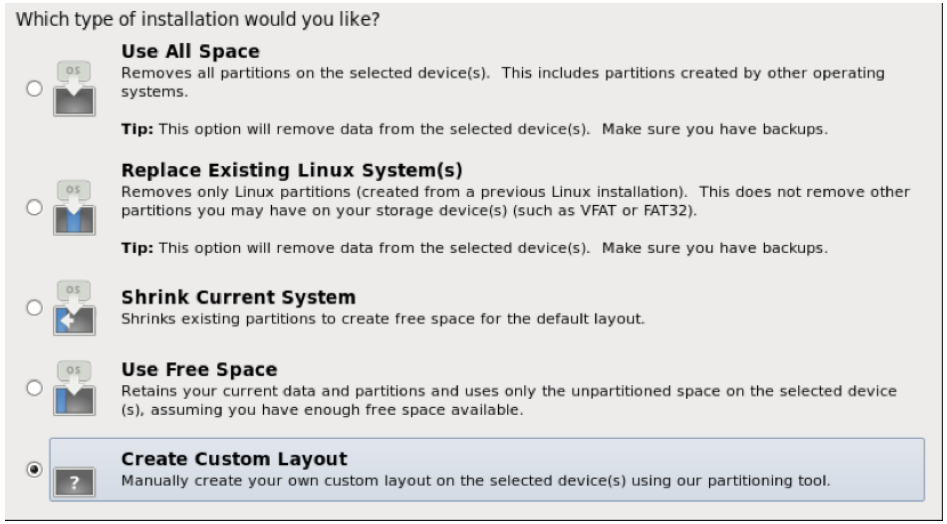
Chapter 19: Creating Partitions

Introduction to creating partitions

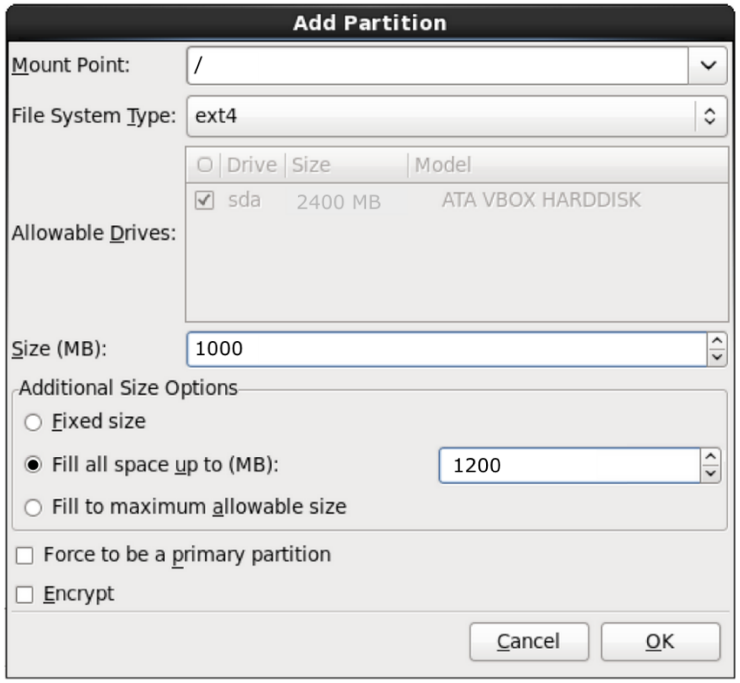
- The creation of partitions can be accomplished either during the installation process or anytime afterwards
  - Typically creating the partitions during the installation is easier because most installation programs provide a GUI-based program
  - The technique to create partitions after the installation is a CLI-based program

Creating Partitions During Installation



Typical initial installation screen

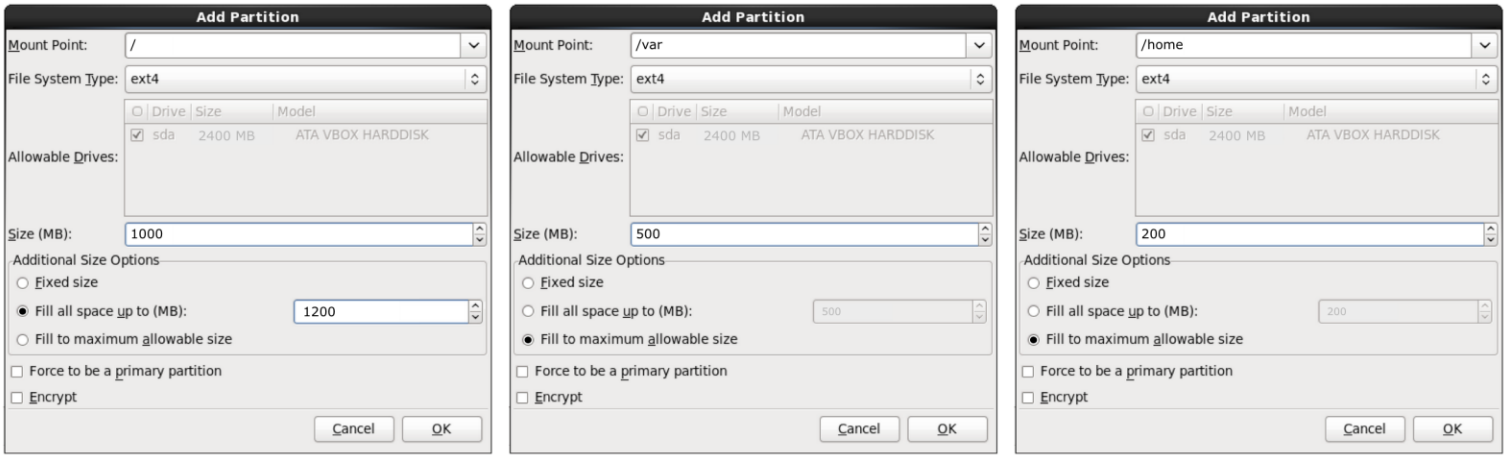
Typical Custom Layout screen



Creating Partitions During Installation

- Example partitioning of a 2400MB Disk:

Partition	Mount Point	Size	Additional Size
Partition1	/	1000 MB	Fill all space up to 1200 MB
Partition2	/var	500 MB	Fill to maximum allowable size
Partition3	/home	200 MB	Fill to maximum allowable size



Creating Partitions After Installation

- The `fdisk` command allows you to create and delete partitions after the installation process
- An interactive tool that doesn't make changes until you "save"
- Use the `-u` option to display in sectors
- Use the `-c` option to avoid MS-DOC compatibility warnings

Display partition information

- Use `fdisk -l` to list existing partitions on a device:

```
# fdisk -cul
Disk /dev/sda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000571a2

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        2481    19921920   83  Linux
/dev/sda2            2481        2611     1046529    5  Extended
/dev/sda5            2481        2611     1046528   82  Linux swap / Solaris
```

Understanding -l output

- Disk /dev/sda: 21.5 GB, 21474836480 bytes - The device name and size of the device
- The Device column - The specific partition that the row is describing
- Start - The starting sector of the partition
- End - The ending sector of the partition
- Blocks - The size of the partition in blocks

Understanding -l output

- Id - An identifier which is used to tell the kernel what time of filesystem should be placed on this partition.
- System - A human-readable name that indicates what type of filesystem the Id column refers to. For example, 83=Linux.

Fdisk Interactive Mode

- The `fdisk` command without –`l` provides you with an interactive method to create or modify partitions
- Use “m" at prompt for a menu of options

```
# fdisk -cu /dev/sda
Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Modifying partitions

- The `fdisk` command without `-l` provides you with an interactive method to create or modify partitions
- Use "p" at prompt to print partition information
- Use "n" to create new partition then answer:
  - Type: e (extended) or p (primary)
  - Partition #
  - Starting sector
  - Partiton size

Example of creating partition

```
# fdisk -cu /dev/sda
Command (m for help): p
Disk /dev/sda: 11.3 GB, 11261706240 bytes
255 heads, 63 sectors/track, 1369 cylinders, total 21995520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ee7d2

 Device Boot      Start         End      Blocks           Id   System
/dev/sda1   *        2048     1026047      512000     83  Linux
/dev/sda2         1026048    20971519     9972736 8e  Linux LVM

Command (m for help): n
Command Action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First sector (20971520-21995519, default 20971520):
Using default value 20971520
Last sector, +sectors, or +size{K,M,G} (20971520-21995519, default 21995519): +100M
```

Modifying partitions

- Use "t" to change partition type

  Enter L to display a list:

```
Hex code (type L to list codes): L
0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
1  FAT12           39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
2  XENIX root      3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
3  XENIX usr       40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
4  FAT16 <32M      41  PPC PReP Boot   85  Linux extended  c7  Syrinx
5  Extended        42  SFS             86  NTFS volume set da  Non-FS data
6  FAT16           4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
7  HPFS/NTFS       4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
```

- Use "d" to delete a partition
- Use "q" to quit without saving changes
- Use "w" to save changes and quit

Saving Partition Changes

- Double check partition information before saving
- Use the w (write) command from menu to write table changes to the MBR and exit
- Use q (quit) to discard changes

sfdisk Command

- Scriptable fdisk-like program for automate partitioning
- Also used to backup and restore current partition table
- sfdisk options:
  - -Use –s to list the disk(s) and sizes(s)
  - -Use –d to backup current table before making changes
  - -Use –f to restore the original table in the event a mistake is made while using partition editing tools

Logical Volume Management

- Logical Volume Management (LVM) is a method of managing hard disk space that provides more flexibility than traditional partitioning of disks
- Supports adding and removing disks by hot plugging and without downtime
- Supports using snapshots for backups of live filesystems

Logical Volume Management

- LVM is implemented by grouping one or more physical volumes into a volume group
- Physical volumes can be normal partitions or even entire disks
- Space from the volume group can be allocated to one or more logical volumes, which are used like normal partitions

Advantages of LVM

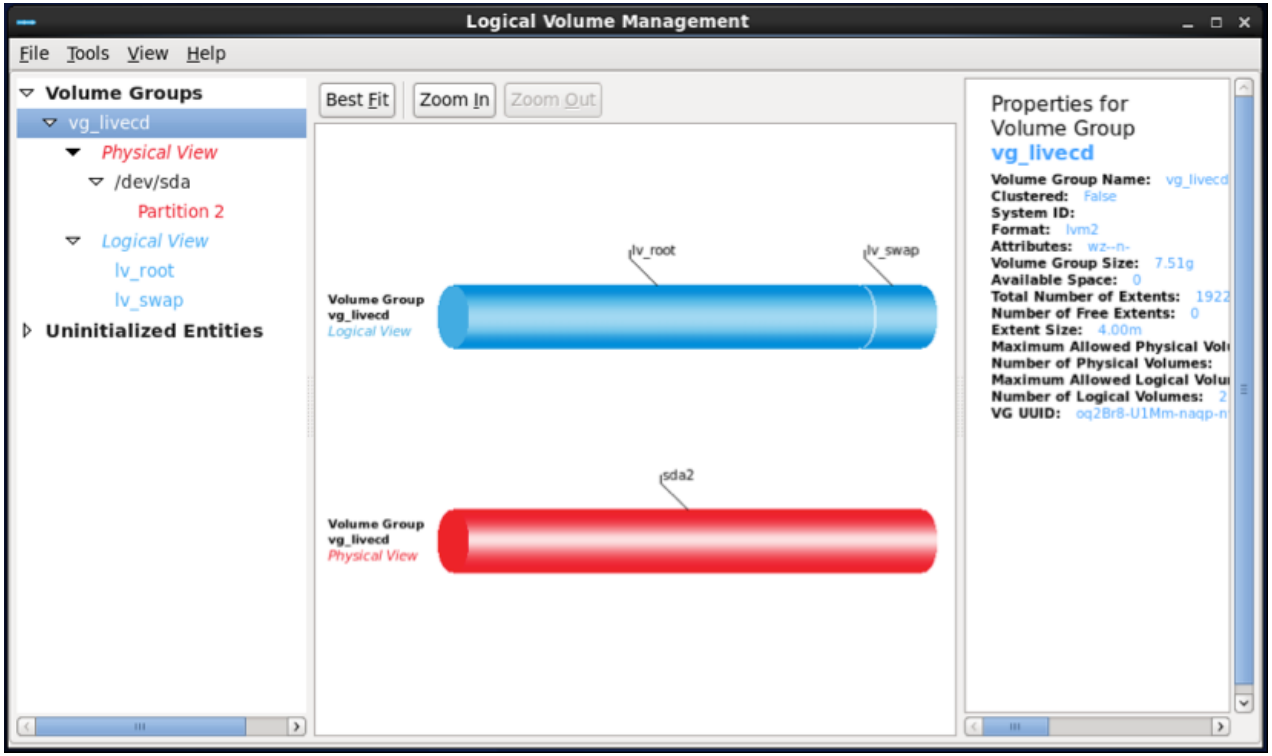
- Resize "live" filesystem using `lvextend` and `resize2fs`
- Add additional space to Volume Group with `vgextend` command

Creating LVM

- Use the `pvcreate` command to create one or more physical volumes from disk partitions or entire disks
- Use the `vgcreate` command to name and create a volume group with one or more of the previously created physical volumes
- Use the `lvcreate` command to name and allocate space to a logical volume

LVM GUI Tools

- Some distributions, like Red Hat, provide a GUI-based tool to administer LVM



LVM device names

- Real logical volume device names follow the pattern `/dev/dm0`, `/dev/dm1`, etc.
- Two symbolic links to each logical volume device name are made:
  - One in the `/dev/mapper` directory
  - One in a subdirectory of `/dev` named after the volume group name

LVM symbolic links

- For volume group named vg_livecd and two logical volumes lv_root and lv_swap:

```
/dev/mapper/vg_livecd-lv_root
/dev/mapper/vg_livecd-lv_swap
/dev/vg_livecd/lv_root
/dev/vg_livecd/lv_swap
```

Creating Filesystems

- The `mkfs` command creates filesystems
  - Use `-t` to specify filesystem type:

    #mkfs –t vfat /dev/sdb1
- The `mkfs` command is a front-end program to other commands, like `mkdosfs`

Common Filesystems

Name	Type Code	Advantages	Disadvantages
Fourth Extended Filesystem	ext4	Support for very large disk volumes and file sizes. Can operate with or without a *journal*. Backwards compatible with ext3 and ext2.	Not a huge improvement over ext3. No dynamic inode creation.
Third Extended Filesystem	ext3	Upgrading from ext2 can be done in-place. This filesystem performs journaling, which allows for rapid recovery after a crash.	The size of volumes and filesystems supported is smaller than ext4.
Second Extended Filesystem	ext2	Works well with small and solid-state disk filesystems. Writes less to disk than ext3.	No journaling capability.

Common Filesystems

Name	Type Code	Advantages	Disadvantages
The Extended Filesystem	ext	Allowed for larger files and filesystems than the Minix filesystem.	Obsolete and no longer supported.
The Minix Filesystem	minix	Borrowed from the academic Minix operating system, this was the first file system supported by Linux.	Obsolete and no longer supported.
The Reiser Filesystem	reiserfs	The first journaling filesystem for Linux. Works efficiently with small files.	Development of this version has ceased with its successor Resiser4 proceeding slowly without help from the founder.

Common Filesystems

Name	Type Code	Advantages	Disadvantages
Journaled Filesystem	jfs	Works well under a wide range of conditions. Uses a journal for faster recovery. Compatible with AIX and OS/2 operating systems from IBM.	Performance is not as good as ext4 under most conditions.
Extents Filesystem	xfs	Works very efficiently with large files. Compatible with the IRIX operating system from SGI. Announced to be default filesystem for RHEL 7.	The filesystem cannot be shrunk.
B-tree Filesystem	btrfs	Includes many advanced features including dynamic inode creation and extent-based file storage.	The program code for btrfs is currently under heavy development.

Common Filesystems

Name	Type Code	Advantages	Disadvantages
File Allocation Table	vfat	Supported by almost all operating systems. Commonly used for removable media.	Unable to support very large disks or files.
New Technology File System	ntfs	Default filesystem used by Windows NT™ family of operating systems.	Driver support is reverse engineered and does not implement all features.

Common Filesystems

Name	Type Code	Advantages	Disadvantages
ISO 9660	iso	The International Organization for Standardization standard for optical disc media that is supported by almost all operating systems.	Multiple levels and extensions complicate compatibility. Not designed for rewritable media.
Universal Disc Format	udf	Designed to replace ISO 9660 and adopted as the standard format for DVDs by the DVD Consortium.	Write support is limited to support revision 2.01 of the standard.

The mke2fs command

- Commonly executed indirectly by the mkfs command when ext2, ext3 or ex4 filesystems are created
- Has several important options:

Option	Description
-b	Specifies the block size of the filesystem.
-N	Specifies the number of inodes.
-m	Specifies what percentage of the filesystem is reserved for system use.

Creating Swap Space

- Swap space is hard drive space set aside to store data normally stored in RAM
- When RAM becomes full, the kernel swaps inactive data and swaps it to the hard drive
- Also called virtual memory

Creating Swap Space

1. Create a partition of type “82” with `fdisk`
2. To initialize a swap partition:

   #`mkswap /dev/sda1` or (to add label)

   #`mkswap -L myswap /dev/sda1`
3. Use `swapon` to enable swap space:

   # `swapon /dev/sdb1`

Creating a Swap File

- Create when there is no unpartitioned space left on the disk
- Steps:
  1. Use `dd` to create a large swap file:

```
#dd if=/dev/zero of=swapfile bs=1M
 count=100
```

  2. Initialize a swap partition:

```
#mkswap swapfile
```

  3. Enable swap space:

```
#swapon swapfile
```

Swap Space Command Summary

- Use `swapon` to enable swap space:

```
# swapon /dev/sdb1
# swapon swapfile
```

- Use `swapon -s` to view swap space
- Use `swapoff` to view swap space

```
# swapoff /dev/sdb1
# swapoff swapfile
```