Cheatography

Linux

by jhaley32 via cheatography.com/123058/cs/23103/

Adding a new blk device

1. Device discovery	follow dmesg, or fdisk -l, of lsblk. find location or new device in /dev/
2. format device to create partition	fdisk, gpart; several options.
3. create a file system for the newly made partition	mkfs -t ext4 /dev/sda*
4. create mount dir	mkdir *mount directory*
5. mount drive	mount /dev/sda mount directory*
6. add entry into fstab	try using blkid to find UUID, then add device with UUID to /etf/fstab. UUID; mount point; FS type; options; backup infomation(0); FS integrity test order

Process Signals

Signal	Portable Number	Default Action	Description
SIGHUP	1	Terminate	Hangup
SIGINT	2	Terminate	Terminal interrupt signal
SIGQUIT	3	Terminate (core dump)	Terminal quit signal
SIGILL	4	Terminate (core dump)	Illegal instruction

Process Signals (cont)

SIGTRAP	5	Terminate (core dump)	Trace/breakpoint trap
SIGABRT	6	Terminate(core dump)	Process Abort Signal
SIGFPE	8	Terminate (core dump)	Erroneous arithmetic operation
SIGKILL	9	Terminate	Kill (cannot be caught or ignored)
SIGSEGV	11	Terminate (core dump)	Invalid memory reference
SIGPIPE	13	Terminate	Write on a pipe with no one to read it
SIGALARM	14	Terminate	Alarm Clock
SIGTERM	15	Terminate	Termination signal

Inode breakdown

Size of File	Size of the file
Device ID	
UID	User ID
GID	Group ID
TIMESTAMPS	access, modify, change-(inode)
MODE	permissions
12 Direct Pointers	points to first 12 data blocks of the file
Indirect	points to a table of addresses for next blocks in a file
x2 Indirect	points to a series of tables with extrea data blocks,

Inode breakdown (cont)

x3 Indirect	same as above x3

systemctl

starting and stopping services	
`sudo systemctl start application.service`	
`sudo systemctl start application`	systemctl looks for *.service already, not necessary to actually place in command.
`sudo systemctl stop application.service`	
restarting and reloading	
`sudo systemctl restart application.service`	fully restart srevice.
`sudo systemctl reload application.service`	reload config files without restarting.
`sudo systemctl reload-or-restart application.service`	if unsure that it can reload, this will try reload first then restart.
Enabling and Disabling Services	
`sudo systemctl enable application.service`	creates a sym link from the system's copy of the service file (usually in `/lib/systemd/system` or `/etc/systemd/system`) into the location on disk where systemd looks for autostart files.
`sudo systemctl disable application.service`	removes sym link

systemctl (cont)

Checking the Status of Services

`systemctl status application.service`	This will provide you with the service state, the cgroup hierarchy, and the first few log lines.
`systemctl is-active application.service`	This will return the current unit state, which is usually active or inactive. exit code 0 if true
`systemctl is-enabled application.service`	This will output whether the service is enabled or disabled. Exit code 0 if true.
`systemctl is-failed application.service`	This will return active if it is running properly or failed if an error occurred. If the unit was intentionally stopped, it may return unknown or inactive. Exit code `0` if failure has occurred.

Listing Current Units

`systemctl list-units`	This will show you a list of all of the units that systemd currently has active on the system

systemctl (cont)

`systemctl list-units --all`	This will show any unit that systemd loaded or attempted to load, regardless of its current state on the system.
systemctl list-units --all --state=inactive	Another common filter is the --type= filter. We can tell systemctl to only display units of the type we are interested in.
`systemctl list-unit-files`	Units are representations of resources that systemd knows about. Since systemd has not necessarily read all of the unit definitions in this view, it only presents information about the files themselves.

Displaying a Unit File

`systemctl cat atd.service`	The output is the unit file as known to the currently running systemd process.

Displaying Dependencies

systemctl (cont)

`systemctl list-dependencies sshd.service`	This will display a hierarchy mapping the dependencies that must be dealt with in order to start the unit in question.

Checking Unit Properties

`systemctl show sshd.service`	To see the low-level properties of a unit, you can use the show command.
systemctl show sshd.service -p Conflicts	To display a single property, you can pass the -p flag with the property name.
Masking and Unmasking Units	mark a unit as completely unstartable, automatically or manually, by linking it to /dev/null.
`sudo systemctl mask nginx.service`	This will prevent the Nginx service from being started, automatically or manually, for as long as it is masked.
`sudo systemctl unmask nginx.service`	This will return the unit to its previous state, allowing it to be started or enabled.

Editing Unit Files

By **jhaley32**
cheatography.com/jhaley32/

Published 4th June, 2020.
Last updated 4th June, 2020.

systemctl (cont)

`sudo systemctl edit nginx.service`	This will be a blank file that can be used to override or add directives to the unit definition. A directory will be created within the /etc/systemd/system directory which contains the name of the unit with .d appended. For instance, for the nginx.service, a directory called nginx.service.d will be created.
`sudo systemctl edit --full nginx.service`	This will load the current unit file into the editor, where it can be modified.
`sudo rm -r /etc/systemd/system/nginx.service.d`	remove snippet created
`sudo rm /etc/systemd/system/nginx.service`	To remove a full modified unit file, we would type:

Adjusting the System State (Runlevel) with Targets

`systemctl get-default`	find the default target for your system

systemctl (cont)

`sudo systemctl set-default graphical.target`	If you wish to set a different default target, you can use the set-default. For instance, if you have a graphical desktop installed and you wish for the system to boot into that by default, you can change your default target accordingly:

Listing Available Targets

systemctl list-unit-files --type=target	You can get a list of the available targets on your system by typing
systemctl list-units --type=target	To see all of the active targets, type:

Isolating Targets

```
systemctl list-dependencies multi-user.target
```

```
sudo systemctl isolate multi-user.target
```

Using Shortcuts for Important Events

`sudo systemctl rescue`	put the system into rescue (single-user) mode, you can just use the rescue command
`sudo systemctl halt`	To halt the system, you can use the halt command:
`sudo systemctl poweroff`	To initiate a full shutdown, you can use the poweroff command:

systemctl (cont)

`sudo systemctl reboot`	A restart can be started with the reboot command: