Using Linux Manual Pages

Learning Goals:

Learn how to locate and read documentation using the built-in manual (man) pages on a Linux system.

Overview of Linux Man Pages

Linux systems include a comprehensive1 set of local documentation files called manual pages or man pages. These documents are installed with most software packages and are accessible directly from the terminal using the man command. By default, these files reside in the /usr/share/man directory, organized by category.

Man pages trace their origins to the early Unix Programmer's Manual, which due to its size was divided into several numbered sections—each dedicated to a specific type of content.

Common Sections in the Manual System

Section	Content Type	Description
1	User Commands	Programs and command-line utilities
2	System Calls	Functions provided by the Linux kernel
3	Library Functions	APIs and routines from standard libraries
4	Special Files	Details on device files and drivers
5	File Formats and Config Files	Syntax and structure of configuration files
6	Games and Screensavers	Legacy games and entertainment programs
7	Miscellaneous, Protocols, Standards	Conventions, protocols, and other non-categorized topics
8	Administrative Commands	Tools used for system management and maintenance
9	Kernel Development	Linux kernel internals and APIs (for developers)

When the same name exists in multiple sections, the reference includes the section number in parentheses to clarify—e.g., passwd(1) covers the user command, while passwd(5) refers to the password file format.

How to View Specific Man Pages

Use man [topic] to display the documentation for a specific command or topic. If a topic exists in more than one section, the man command defaults to the first matching section (typically section 1). To view a page from a specific section, use:

```
$ man [section] [topic]
```

Example:

```
$ man 5 passwd
```

Sections most relevant to system administrators are:

- **1: Commands**
- **5: Configuration files**
- **8: System utilities**

Sections 2 and 3 are especially useful for developers.

Navigating Through Man Pages

Being able to navigate man pages efficiently is crucial for working from the command line. Here are common keybindings used when browsing:

Key / Command	Action
`Spacebar`	Scroll down one page
`PageDown`	Scroll down one page
`PageUp`	Scroll up one page
`↓ / ↑`	Scroll line-by-line
`D / U`	Scroll down/up half a page
`/string`	Search forward for a keyword
`n`	Repeat last search (forward)
`N`	Repeat last search (backward)
`G`	Jump to the top of the page
`Shift+G`	Jump to the end of the page
`q`	Quit and return to the terminal

You can search using regular expressions, though be cautious with special characters like ^, ., *, or $, which may produce unexpected results.

For more on regex syntax, refer to:

```
$ man 7 regex
```

Structure of a Typical Man Page

Each man page usually follows a consistent layout with common headers. These help users quickly find the information they need:

Header	Description
`NAME`	Command or topic name with a brief summary
`SYNOPSIS`	Command syntax or function prototype
`DESCRIPTION`	Overview or purpose of the command/topic
`OPTIONS`	List and explanation of command-line options
`EXAMPLES`	Practical usage examples
`FILES`	Related configuration or system files
`SEE ALSO`	Related commands or documentation
`BUGS`	Known issues
`AUTHOR`	Contributor or maintainer information

Finding Man Pages by Keyword

To search for relevant man pages using a keyword, use:

```
$ man -k [keyword]
```

This is equivalent to the apropos command and searches the short descriptions of available topics.

Example:

```
$ man -k passwd
```

Output might include:

```
passwd (1)      - update user's authentication tokens
passwd (5)      - password file
chpasswd (8)    - update passwords in batch mode
```

For a deeper search through the entire content of man pages, use the -K option (note the uppercase K):

```
$ man -K [keyword]
```

This performs a full-text search but is resource-intensive. As it finds matches, it prompts whether to view, skip, or quit. For example:

```
--Man-- next: chage(1) [ view (return) | skip (Ctrl-D) | quit (Ctrl-C) ]
```

You can:

- Press Enter to read the match.
- Press Ctrl+D to skip to the next match.
- Press Ctrl+C to exit the search.

Note: Full-text and keyword searches depend on an up-to-date man page index created by the mandb command, which is usually triggered automatically when new packages are installed.

Further References

To learn more, explore:

- man(1)
- mandb(8)
- man-pages(7)
- less(1)
- intro(1), intro(2), intro(5), intro(7), intro(8)

Understanding the System Log Framework

Overview of Linux Logging Mechanisms

When a Linux system is active, both the kernel and various running processes generate event logs. These logs play a crucial role in auditing system activity and diagnosing technical issues. Command-line tools like less and tail are commonly used to examine these log files directly.

Red Hat Enterprise Linux (RHEL) employs a logging framework based on the syslog protocol to manage and store these system messages. In RHEL 9, two primary services are responsible for this logging functionality: systemd-journald and rsyslog.

Role of systemd-journald

The systemd-journald service is the core component of the RHEL logging infrastructure. It gathers log entries from a variety of sources, including:

- Kernel messages
- Boot process outputs
- Standard output and error from system services
- Traditional syslog entries

Once collected, systemd-journald converts these messages into a standardized format and stores them in an indexed system journal. By default, this journal resides in memory and is non-persistent, meaning its contents are lost after a system reboot.

Role of rsyslog

The rsyslog service supplements systemd-journald by reading its logs in real-time. It then:

- Filters and processes the logs
- Writes them to persistent log files
- Organizes them based on message priority and source

These logs are written to the /var/log directory, ensuring they remain available across system restarts.

Key Log Files in /var/log

The /var/log directory contains multiple log files generated by various services. Below are some notable entries:

Log File	Purpose
`/var/log/messages`	General system messages, excluding authentication, mail, and scheduled jobs
`/var/log/secure`	Authentication and security-related messages
`/var/log/maillog`	Mail server logs
`/var/log/cron`	Details about scheduled (cron) job execution
`/var/log/boot.log`	Messages related to the system boot process (not from syslog)

Non-syslog Logging Applications

Not all software uses the syslog pipeline. For example, Apache HTTP Server maintains its own logging structure, typically under /var/log/httpd/ or a similar path depending on your configuration.

Further Reading & Resources

To explore the logging system in more depth, consult the following manual pages:

- **systemd-journald.service(8)**
- **rsyslogd(8)**
- **rsyslog.conf(5)**

You can also refer to the official RHEL 9 guide on basic system settings here:
Red Hat Enterprise Linux 9 – Configuring Basic System Settings

Understanding Syslog Files

Learn how to read and interpret syslog files for troubleshooting and system monitoring.

System Event Logging

On a Red Hat Enterprise Linux system, many programs report their activities by sending log messages using the syslog protocol. Each of these messages is categorized based on two elements:

- **Facility: The source subsystem that generated the message.**
- **Priority: The seriousness of the message.**

Syslog Facilities Overview

Syslog facilities identify where a log message originates. Here's a quick summary:

Code	Facility	Description
0	kern	Kernel-related messages
1	user	Messages generated by user applications
2	mail	Mail service messages
3	daemon	System daemon processes
4	auth	Authentication and security logs
5	syslog	Internal syslog service logs
6	lpr	Printer system messages
7	news	Network news services
8	uucp	UUCP protocol events
9	cron	Scheduled job logs
10	authpriv	Secure authentication messages
11	ftp	FTP service logs
16–23	local0-local7	Custom application logs

Syslog Priorities Overview

Priorities rank the importance of events, from most critical to least:

Code	Priority	Description
0	emerg	System is unusable
1	alert	Immediate action required
2	crit	Critical issues
3	err	Error conditions
4	warning	Warning messages
5	notice	Normal but noteworthy events
6	info	General information
7	debug	Debugging information

How rsyslog Handles Log Messages

The rsyslog service uses rules defined in /etc/rsyslog.conf and /etc/rsyslog.d/*.conf files to decide how to process log entries based on their facility and priority.
Rules are written like this:

```
facility.priority   /path/to/logfile
```

The asterisk (*) can serve as a wildcard matching any facility or priority.

Example:
To capture all authentication events into /var/log/secure, you'd add:

```
authpriv.*   /var/log/secure
```

If multiple rules match a message, it can be stored in several places unless you use the none keyword to explicitly exclude it. Critical emergency (emerg) messages can even be broadcasted to all logged-in users' terminals.

Example rsyslog Rules

```
# Kernel messages to console
kern.*

# General logging, excluding mail, authpriv, and cron
*.info;mail.none;authpriv.none;cron.none /var/log/messages

# Secure authentication logs
authpriv.* /var/log/secure

# Mail service logs
mail.* -/var/log/maillog

# Scheduled tasks logs
cron.* /var/log/cron

# Emergency messages to all users
*.emerg :omusrmsg:*

# Network news and UUCP critical logs
uucp,news.crit /var/log/spooler

# Boot messages
local7.* /var/log/boot.log
```

Note: Syslog offers many more advanced features. You can explore them by checking the rsyslog.conf(5) manual page or browsing

/usr/share/doc/rsyslog/html/index.html if the rsyslog-doc package is installed.

Managing Log Files with logrotate

Log files in /var/log can quickly grow large. To manage disk usage, the logrotate tool automatically rotates these files.
When a file rotates, it's renamed with a date suffix (e.g., /var/log/messages-20250420).

Typically, log files are rotated weekly, but some may rotate sooner if they reach a certain size. Old logs are eventually deleted, keeping only a few weeks' worth.

Reading and Analyzing Log Entries

Logs record entries in a chronological order—older messages at the top and newer ones at the bottom.

Here's an example entry from /var/log/secure:

```
Mar 20 20:11:48 localhost sshd[1433]: Failed password for student from 172.25.0.10 port 59344 ssh2
```

Breaking it down:

- Mar 20 20:11:48 – Timestamp of the log
- localhost – Hostname
- sshd[1433] – Process name and its PID
- Failed password for student... – Message content

Monitoring Logs in Real-Time

To watch log files live, use:

```
tail -f /var/log/secure
```

This will display new log lines as they are written.
You can, for example, run an SSH login attempt on another terminal to watch authentication attempts in real-time.

Manually Sending Log Entries

You can manually send messages into the system log using the logger command:

```
logger -p local7.notice "Manual log entry on host"
```

This command logs the message with the local7 facility and notice priority—useful for testing logging setups.

References:

- logger(1), tail(1), rsyslog.conf(5), logrotate(8) man pages
- [Troubleshooting with Log Files - RHEL 9 Documentation]()
- /usr/share/doc/rsyslog/html/index.html (requires rsyslog-doc package)

Exploring System Journal Entries

Learn how to locate and interpret system journal entries to assist with troubleshooting and monitoring system activity.

Locating Events in the System Journal

The systemd-journald service collects and saves log messages in a structured, indexed binary format known as the journal. This format allows additional metadata to be stored alongside each event. For instance, with syslog events, details such as message priority and the associated facility (which identifies the source process) are recorded.

Note:

On Red Hat Enterprise Linux, journal entries are initially stored in the memory-backed /run/log directory. As a result, these logs are cleared during a system shutdown. You can configure journald to store logs persistently, which will be discussed later.

You can access journal logs using the journalctl command. This tool allows you to view the full log history or filter for specific messages based on different criteria. Running journalctl as the root user provides unrestricted access, while regular users may have limited visibility.

Example basic usage:

```
[root@host ~]# journalctl
```

Sample output:

```
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on
PipeWire Multimedia System Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Starting Cre-
ate User's Volatile Files and Directories...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target
Basic System.
...
```

Within the output, important notices and warnings appear in bold, and critical errors display in red for easier identification.

Managing Output with Options

When reviewing system logs, it's crucial to narrow down the output to focus on relevant entries.

- Show recent log entries:
  By default, journalctl -n displays the last 10 events. You can specify a different number if needed:
  ```
  [root@host ~]# journalctl -n 5
  ```
  Output example:
  ```
  Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished
  Cleanup of User's Temporary Files and Directories.
  Mar 15 05:01:01 host.lab.example.com CROND[2197]: (root) CMD
  (run-parts /etc/cron.hourly)
  ...
  ```

- Monitor logs in real-time:
  Similar to tail -f, the -f option lets you continuously watch new log entries as they happen:
  ```
  [root@host ~]# journalctl -f
  ```
  Exit real-time viewing with Ctrl+C.

Filtering by Priority Level

For troubleshooting, you can filter logs by severity using the -p option. journalctl supports the following priority levels, from lowest to highest urgency: debug, info, notice, warning, err, crit, alert, and emerg.

Example: Display only errors and more severe messages:

```
[root@host ~]# journalctl -p err
```

Sample output:

```
Mar 15 04:22:00 host.lab.example.com pipewire-pulse[1640]:
pw.conf: execvp error 'pactl': No such file or directory
Mar 15 04:22:20 host.lab.example.com smartd[669]: DEVICESCAN
failed: glob(3) aborted matching pattern /dev/discs/disc*
...
```

Viewing Logs for Specific Services

You can narrow down logs to a particular systemd unit using the -u option.

Example: Reviewing logs for the SSH daemon:

```
[root@host ~]# journalctl -u sshd.service
```

Output example:

```
May 15 04:30:18 host.lab.example.com systemd[1]: Starting OpenSSH
server daemon...
May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening
on 0.0.0.0 port 22.
...
```

Searching Logs by Time

You can restrict the log output to a specific time range using the --since and --until options. Timestamps must be quoted if they include spaces (e.g., "YYYY-MM-DD hh:mm:ss").

- View today's logs:
  ```
  [root@host ~]# journalctl --since today
  ```

  Sample output:
  ```
  Mar 15 05:04:20 host.lab.example.com sshd[2255]:
  pam_unix(sshd:session): session opened for user stu-
  dent(uid=1000)
  ...
  ```

- View logs between specific dates:
  ```
  [root@host ~]# journalctl --since "2022-03-11 20:30" --until
  "2022-03-14 10:00"
  ```

  This displays all journal entries recorded between March 11 at 8:30 PM and March 14 at 10:00 AM.

  By effectively using journalctl options, you can dramatically speed up system troubleshooting and gain precise insights into system behavior.

Ensuring System Journal Persistence

Learn how to configure Red Hat Enterprise Linux to retain journal logs across system reboots.

Where the System Journal is Stored

By default, on RHEL 9, system logs managed by systemd-journald are stored in /run/log, a temporary filesystem that clears when the system powers down.
To make journal logs survive reboots, you need to modify systemd-journald settings in the /etc/systemd/journald.conf file.

Configuring Journal Storage Options

The behavior of journal storage is controlled by the Storage parameter inside /etc/systemd/journald.conf. You can set it to one of the following values:

- persistent: Saves logs to /var/log/journal, ensuring they remain after reboots. If the directory doesn't exist, systemd-journald will create it.
- volatile: Keeps logs in /run/log/journal, which lives only in memory—logs are lost after reboot.
- auto: Defaults to persistent storage if /var/log/journal exists; otherwise, it falls back to volatile storage.
- none: Disables journal storage entirely, although forwarding logs to external systems remains possible.

Advantages and Journal Size Limits

Persistent journals offer quick access to historical logs immediately after boot. However, retention isn't unlimited:

- Journals automatically rotate monthly.
- Storage is capped to 10% of the filesystem or ensures at least 15% free space remains.

These thresholds can be tuned by editing the journald.conf file.

Checking Current Journal Size Limits

You can check how much space your journals occupy and their maximum allowed sizes with:

```
[user@host ~]$ journalctl | grep -E 'Runtime Journal|System
Journal'
```

Example output:

```
Mar 15 04:21:14 localhost systemd-journald[226]: Runtime Journal
(/run/log/journal/UUID) is 8.0M, max 113.3M, 105.3M free.
Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: System
Journal (/run/log/journal/UUID) is 8.0M, max 4.0G, 4.0G free.
```

Tip: The | symbol in the grep command acts as an OR, matching either string.

Steps to Enable Persistent Journals

1. Create the required directory:
   ```
   [root@host ~]# mkdir -p /var/log/journal
   ```
2. Edit the journal configuration:
   Open /etc/systemd/journald.conf and set:
   ```
   [Journal]
   Storage=persistent
   ```
3. Restart the journald service:
   ```
   [root@host ~]# systemctl restart systemd-journald
   ```

Once configured, systemd-journald will create subdirectories inside /var/log/journal, named with long hexadecimal strings. These directories contain .journal binary files.

Example:
```
[root@host ~]# ls /var/log/journal
4ec03abd2f7b40118b1b357f479b3112

[root@host ~]# ls /var/log/journal/4ec03ab-
d2f7b40118b1b357f479b3112
system.journal user-1000.journal
```

Viewing Journals by Boot Sessions

Even with persistent storage, journalctl by default shows logs from both current and previous boots. You can limit the view:

- Show logs from a specific boot:
  ```
  [root@host ~]# journalctl -b 1
  ```
- Show the second previous boot:
  ```
  [root@host ~]# journalctl -b 2
  ```
- List all recognized boots:
  ```
  [root@host ~]# journalctl --list-boots
  ```

Example output:
```
-6 27de... Wed 2022-04-13 20:04:32—Wed 2022-04-13 21:09:36
-5 6a18... Tue 2022-04-26 08:32:22—Thu 2022-04-28 16:02:33
...
0 ee2c... Mon 2022-05-09 09:56:45—Mon 2022-05-09 12:57:21
```

- View only the current boot:
  ```
  [root@host ~]# journalctl -b
  ```

Important:

When investigating system crashes, it's often helpful to focus on the journal from the boot session prior to the crash. You can specify previous boots with negative numbers, for example:
```
# journalctl -b -1
```

References

- systemd-journald.conf(5)
- systemd-journald(8)

Keeping System Time Accurate

Ensure reliable timekeeping by using Network Time Protocol (NTP) for synchronization and properly setting the system time zone for accurate timestamps across system logs and journal entries.

Managing Local Time and Time Zones

Accurate system time is critical when diagnosing issues across multiple servers. Many system services also rely on precise time synchronization to operate correctly.
Linux systems typically use NTP to synchronize time either with public servers (like the NTP Pool Project) or with a highly accurate local hardware clock.

You can check the current time settings using the timedatectl command:

```
[user@host ~]$ timedatectl
```

Example output:

```
Local time: Wed 2022-03-16 05:53:05 EDT
Universal time: Wed 2022-03-16 09:53:05 UTC
RTC time: Wed 2022-03-16 09:53:05
Time zone: America/New_York (EDT, -0400)
System clock synchronized: yes
NTP service: active
RTC in local TZ: no
```

Exploring Available Time Zones

To view all available time zones, use:

```
[user@host ~]$ timedatectl list-timezones
```

Sample entries:

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
...
```

Linux systems use the IANA Time Zone Database, organizing entries by continent (or ocean) and usually the largest city in that zone (e.g., America/Denver for the US Mountain Time zone).

Some regions differ within the same time zone. For example, while Arizona lies geographically within the Mountain time zone, it doesn't observe daylight saving time and instead uses America/Phoenix.

To interactively determine the correct time zone for your location, you can use:

```
[user@host ~]$ tzselect
```

tzselect helps identify your correct time zone but does not apply it automatically.

Changing the System Time Zone
To change the system's time zone, the root user can run:

```
[root@host ~]# timedatectl set-timezone America/Phoenix
```

Verifying the change:

```
[root@host ~]# timedatectl
```

Example output:

```
Local time: Wed 2022-03-16 03:05:55 MST
Universal time: Wed 2022-03-16 10:05:55 UTC
Time zone: America/Phoenix (MST, -0700)
```

Note:
If you want to set the time zone to Coordinated Universal Time (UTC), use:

```
timedatectl set-timezone UTC
```

Setting the System Time Manually
You can manually set the system time using:

```
[root@host ~]# timedatectl set-time "09:00:00"
```

However, if NTP synchronization is active, you might encounter an error:
"Failed to set time: Automatic time synchronization is enabled."

To disable NTP before setting time manually:

```
[root@host ~]# timedatectl set-ntp false
```

Managing NTP Synchronization

Use timedatectl set-ntp to enable or disable NTP:

```
[root@host ~]# timedatectl set-ntp true   # Enable NTP
[root@host ~]# timedatectl set-ntp false  # Disable NTP
```

Important:
On RHEL 9, this controls the chronyd service behind the scenes. Other distributions may link it to different time synchronization services like ntpd or systemd-timesyncd.

Changing NTP settings via graphical tools (like GNOME Settings) updates the same configuration.

Configuring and Monitoring the chronyd Service

The chronyd daemon manages the system clock, even compensating for hardware clock drift when offline.
By default, it synchronizes time using servers from the NTP Pool Project.

If you need to configure different servers, edit /etc/chrony.conf:

Example server entry:

```
server classroom.example.com iburst
```

Tip:

Adding iburst speeds up the initial synchronization by sending multiple quick requests.

After modifying /etc/chrony.conf, restart the chronyd service:

```
[root@host ~]# systemctl restart chronyd
```

Verifying NTP Synchronization

You can monitor the system's NTP status using chronyc, the command-line client for chronyd:

```
[root@host ~]# chronyc sources -v
```

Sample output:

```
MS Name/IP address       Stratum Poll Reach LastRx Last sample
===============================================================================
^* 172.25.254.254            3     6    17     26    +2957ns[+2244ns] +/- 25ms
```

- * means the system is currently synchronized with that server.
- ^ indicates it's a server (not a peer).
- Stratum levels indicate distance from a reference clock (stratum 0).

References

- timedatectl(1)
- tzselect(8)
- chronyd(8)
- chrony.conf(5)
- chronyc(1)