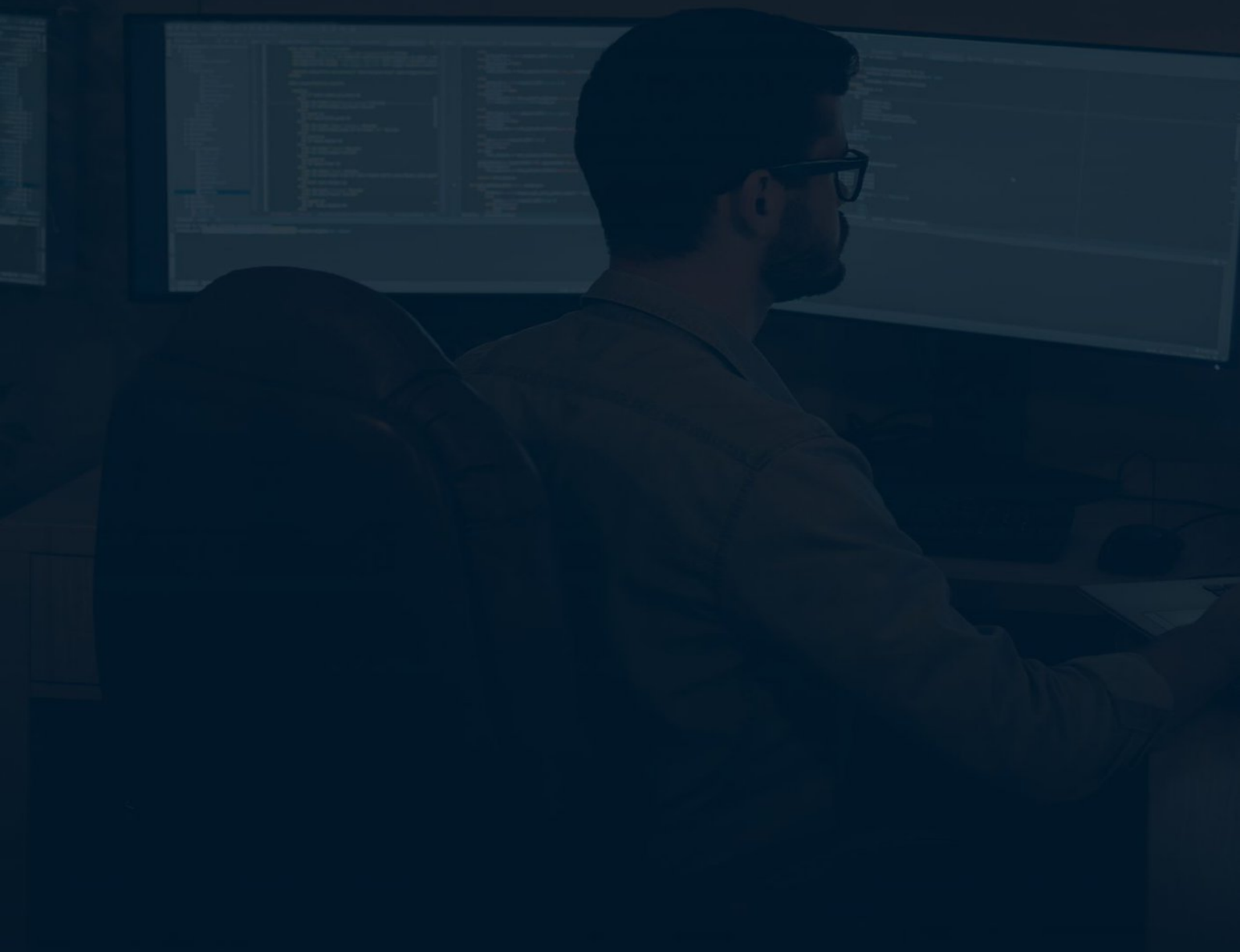
OPEN-SOURCE EBOOK

++101 Linux Commands



BOBBY ILIEV

Table of Contents

The `whatis` command

The `whatis` command is used to display one-line manual page descriptions for commands. It can be used to get a basic understanding of what a (unknown) command is used for.

Examples of uses:

1. To display what `ls` is used for:

```
whatis ls
```

2. To display the use of all commands which start with `make`, execute the following:

```
whatis -w make*
```

Syntax:

```
whatis [-OPTION] [KEYWORD]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-d`	`--debug`	Show debugging messages
`-r`	`--regex`	Interpret each keyword as a regex
`-w`	`--wildcard`	The keyword(s) contain wildcards

The `who` command

The `who` command lets you print out a list of logged-in users, the current run level of the system and the time of last system boot.

Examples

1. Print out all details of currently logged-in users

```
who -a
```

2. Print out the list of all dead processes

```
who -d -H
```

Syntax:

```
who [options] [filename]
```

Additional Flags and their Functionalities

Short Flag	Description
`-r`	prints all the current runlevel
`-d`	print all the dead processes
`-q`	print all the login names and total number of logged on users
`-h`	print the heading of the columns displayed

Short Flag	Description
-b	print the time of last system boot

018-the-free-command.md

The `free` command

The `free` command in Linux/Unix is used to show memory (RAM/SWAP) information.

Usage

Show memory usage

Action: --- Output the memory usage - available and used, as well as swap

Details: --- Outputted values are not human-readable (are in bytes)

Command:

```
free
```

Privacy Considerations

While the `finger` command is useful for retrieving information about system users, it may also expose sensitive details in shared or multi-user environments:

1. **Usernames and Login Times**: Displays login times, which can be used to track user activity.
2. **Home Directories**: Exposes paths to users' home directories.
3. **Idle Status**: Shows how long a user has been inactive, potentially signaling whether they are actively using their system.
4. **Mail Status**: Displays mail information, which may inadvertently reveal user engagement.

Potential Risks:

In environments with untrusted users, the information exposed by `finger` could be exploited for:

- **Social Engineering Attacks**: Malicious actors could use this information to craft personalized phishing attacks.
- **Timing Attacks**: Knowing when a user is idle or active could give attackers an advantage in timing their attempts.
- **Targeted Attacks**: Knowledge of user home directories can focus attacks on those locations.

Mitigating Privacy Risks:

To mitigate these risks, consider limiting access to the `finger` command in environments where user privacy is important.

The `in.fingerd` Service

It’s important to distinguish between the `finger` command and the **`in.fingerd` service**. The `finger` command is local, while `in.fingerd` is a network daemon that allows remote queries of user information. This service is typically disabled by default in modern systems due to potential security risks.

If enabled, the `in.fingerd` service can expose user information over the network, which could be exploited by attackers. To mitigate this risk, system administrators should ensure the service is disabled if it is not needed.

Disabling the `in.fingerd` Service:

If you are concerned about remote queries, you can disable the `in.fingerd` service:

```
sudo systemctl disable in.fingerd
sudo systemctl stop in.fingerd
```

By disabling the `in.fingerd` service, you prevent remote querying of user information, enhancing system security.

The `groups` command

In Linux, there can be multiple users (those who use/operate the system), and groups (a collection of users). Groups make it easy to manage users with the same security and access privileges. A user can be part of different groups.

Important Points:

The `groups` command prints the names of the primary and any supplementary groups for each given username, or the current process if no names are given. If more than one name is given, the name of each user is printed before the list of that user's groups and the username is separated from the group list by a colon.

Syntax:

```
groups [username]
```

Example 1

Provided with a username

```
groups demon
```

In this example, username demon is passed with groups command and the output shows the groups in which the user demon is present, separated by a colon.

Example 2

When no username is passed then this will display the group membership for the current user:

```
groups
```

Here the current user is demon . So when we run the `groups` command without arguments we get the groups in which demon is a user.

Example 3

Passing root with groups command:

```
$demon# groups
```

Note: Primary and supplementary groups for a process are normally inherited from its parent and are usually unchanged since login. This means that if you change the group database after logging in, groups will not reflect your changes within your existing login session. The only options are –help and –version.

This is a sample from "101 Linux Commands eBook" by Bobby Iliev the Hacktoberfest community.

For more information, Click here.