**Cheat Sheet – UCB Cyber Bootcamp**

**Resources**

* [Debian Linux](https://www.debian.org/intro/about)
* [Ubuntu Linux](https://www.ubuntu.com/download)
* [Kali Linux](https://www.kali.org/about-us/)
* [RedHat](https://www.redhat.com/en/technologies)
* [Fedora](https://getfedora.org/)
* [CentOS](https://www.centos.org/about/)
* [SELinux](https://selinuxproject.org/page/Main_Page)
* [Mint Breach](https://www.techrepublic.com/article/why-the-linux-mint-hack-is-an-indicator-of-a-larger-problem/)
* [Linux File System Hierarchy](https://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)

**Key Terms**

* **Operating System**: Also known as an OS, this is a platform that allows users to install and run applications, such as web browsers and text editors.

**Example**: Windows, Mac OS X, and Linux are all examples of operating systems.

* **FOSS**: **free, open source software (FOSS)**. This means that anyone can read or modify its **source code**.

**Example**: Linux is free and open source specifically because Windows and Unix, its early competitor, were not. In other words, Linux was developed for researchers and students in direct response to the fact that Windows and Unix were extremely expensive and/or inaccessible in the early days of computing.

* **Linux Distribution**: Because Linux is FOSS, many different people have developed their own special-purpose variants of the operating system. These variants are called **distributions**.

**Example**: There are many different distributions in use today. In this course, students will use two: **Ubuntu** and **Kali Linux**.

* **Headless Servers**: Most production Linux servers don't even offer a graphical interface—they can *only* be used from the command line. Such command line-only machines are called **headless servers**.

**Example**: Headless servers are the norm because by today's standards, the CLI requires *very* little resources. This gives the server maximum resources to run it's services and applications, so a GUI based system is neither required nor desirable.

* **System-wide Configurations**: While normal users can only modify their own local configurations, an administrator can make changes that apply to *all* users of the system. These are called **system-wide configurations**, and apply to every user on the system.

**Example**: An administrator might configure a machine to prevent people for browsing to https://facebook.com. This would apply to *all* users of the system, making it a system-level configuration.

* **Password Hash**. This is a string that is *different* from the user's password, which the operating system can still use to check that they've entered the correct password to log in.

**Example**: A hash is a cryptographic function that changes the password to an unintelligible string of characters so it is not easily read or used by an attacker. $6$6Y/fI1nx$zQJj6AH9asTNfhxV7NoVgxByJyE.rVKK6tKXiOGNCfWBsrTGY7wtC6Cep6co9eVNkRFrpK6koXs1NU3AZQF8v/ is an example of a hash

* **/etc/passwd**: This file contains a list of registered users on the system. Contrary to its name, it contains no information about user passwords.

**Example**: This means that attackers who steal /etc/passwd will get a list of users to attack, but will *not* be able to steal their passwords.

* **/etc/shadow**: This file contains both a list of usernames *and* information about their passwords.

**Example**: Specifically, it stores the password hashes.

* **System Process**: A program that is *running* is called a **process**, because it is in the *process* of performing its task.

**Example**: When a program runs, it must process data, and potentially make changes to the file system. For example, a text editing application needs to save temporary files to backup users' work, and then save their final draft to a file on disk.

* **PID**: The Process ID. Every running process on the system is assigned a PID by the system, so the system can keep track of which process is which.

**Example**: The PID for a process is shown with commands like top and ps. When you want to stop a process, you would often use the PID along with the kill command. e.g. kill 947

* **/etc/security/pwquality.conf**: The file that contains the rules for all passwords created on the system.

**Example**: Edit this file to require users to create stronger passwords.

* **Package Manager**: To install software on a Linux system, Linux has a program called a **package manager**. Administrators often have no choice but to install additional software to properly harden the machines they manage.

**Example**: The package manager used by Ubuntu is called aptitude. You use aptitude with the command apt.

* **Repository**: When you install a package with apt, Linux searches special databases to find information about <package name>. If it finds it, it will download and install the package. These databases are known as **repositories**.

**Example** Repositories specifically used to store and distribute packages are known as **Personal Package Archives**, or **PPAs**. PPAs are simply servers where Linux software is stored and maintained.

* **Access Controls**: These controls determine that actions that users are able to do to a file (edit, view, etc.).

**Example**: Google Docs is a fantastic example of access controls at work because we can choose who we share files with and what permission they have when they receive the file, such as whether they can only read or make edits to it.

* **Discretionary Access Control**: Also known as DAC. It is called discretionary, because the owner of an item can specify what other users can access the item. In other words, Access control is based on the discretion of the owner.

**Example**: A directory may pass on it's permissions to items inside it.

* **Mandatory Access Control**: Also known as MAC - This differs from DAC because with MAC the system decides what users have access to what items. Each user is given a certain level of clearance that allows them to access certain types of files

**Example**: SELinux is a mandatory access control system.

* **Permissions** regulate *who* can take which actions. For example, Jane can read and write, but John can only read.

**Example**: On a Linux system permissions are set for each file or directory.

* **File Permissions**: The set of 10 permissions flags assigned to every file/directory.

**Example**: -rwxrwxrwx - The first flag represents the file type. If it is a directory, a 'd' is shown. If it is a file, a - is shown. The remaining 9 flags are comprised of 3 groups of rwx. r for read, w for write and x for execute. Each of the 3 sets of letters represent an entity and it's respective read, write, and execute permissions for that file/directory. The first set of rwx represent the owner's permissions. The second set of rwx represent the group's permissions and the third set of rwx represents the permissions of all other users on the system. The letters in each set of rwx never change position. If a certain permission is missing it is represented by a -. e.g. drwxr-xr-- shows a directory where the owner has full permissions to read, write and execute, the group can read and execute but not write, and all other users can only read.

* **File or Directory Owner**: The 'owner' is the main user assigned to a file or directory.

**Example**: The owner of a file/directory is usually the user that created said file/directory. When looking at an file/directory's permissions, the owner's permissions are represented by the first set of rwx listed. When you create a file or directory, you become the owner of that file or directory. If a program creates an file/directory, the owner of the file/directory is not the program, but rather the user that started the program.

* **File or Directory Group**: The 'group' of users or programs assigned to a file or directory.

**Example**: The group of a file or directory represents all users that are a member of that group. In other words, it represents an entire group of users. When viewing an file/directory's permissions, the group's permissions are represented by the second set rwx letters.

* **File or Directory Other/World**: A category that represents all 'other' users on a system.

**Example**: The Other/World category of a file/directory represents any user that is not in the group assigned to the file/directory, and it is not the owner of that file/directory. In other words, it is any other user that is not directly associated with that file/directory. When viewing an file/directory's permissions, the 'Other/World' permissions are represented by the third set rwx letters.

* **SUID**: The Set User ID special permissions bit. This bit is only used on files that have the x bit set in the owner position. In other words, it is only used on executable files or programs.

**Example**: It causes the executable to behave as if the owner executed the file, regardless of what user on the system executed it.

* **SGID**: The Set Group ID special permissions bit. This bit can be used on both directories and executable files that have the x permissions set for the group position. However, SGID is *rarely* used on executable files.

**Example**: When used on a file, the executable behaves as if the owner who executes the file were a member of the file's group even if they are not. When used on a directory, anything created inside said directory will be assigned the same group as the directory instead of being assigned the primary group of the owner.

* **Sticky**: The sticky special permissions bit. The Sticky bit is only used on directories that have the x permission set for the other position.

**Example**: The Sticky bit causes the directory to allow anyone on the system to create files in said directory, and it allows anyone on the system to delete files in said directory, but *only* if those files belong to the user. In other words, a user cannot delete files that belong to other users inside the directory.

* **Symbolic Notation**: When changing an file/directory's permissions the letters r, w, and x are used directly within a command.

**Example**: +x would add execute privileges and -w would remove write privileges.

* **Octal Notation**: When changing a file/directory's permissions numbers are used to represent the respective r, w, and x.

**Example**: Each letter in a group of rwx is assigned a number if you want to set it. r always equals 4, w always equals 2, and x always equals 1. If you don't want to set that permission, it's value is 0. The octal notation is the sum of the values assigned to each letter. If you want to set permissions to only read, the number is 4. If you want to read and write, add the numbers for r and w together and the number is 6. If you want read and execute, the number is 5. rwx would be equal to 7. This way, you can represent the permissions of the user, group and other with just 3 numbers. e.g. 755 would translate to rwxr-xr-x.

* **Service User**: A system user who's sole purpose is to run *one* service. This keeps services from running as the root user and generally keeps the system more secure.

**Example**: A service such as Apache will automatically install it's own user when the package is installed. If a service does not have it's own user, a user should be created for it so it can run without root privileges.

**Key Commands**

**General Commands**

* `ls` List all the items in a directory
* `cd` Change directory
* `mv` Move a file
* `cp` Copy a file
* `less` Read a file with pagination
* `head` View the top 10 lines of a file
* `tail` View the bottom 10 lines of a file
* `>` Redirect the output of a command into a file
* `mkdir` Create a directory
* `rm` Remove a file or a directory
* `whoami` Display the current user name
* `groups` Display the groups for a user
* `man` Open the manual for a command

**ls -l**

List the 'long' form of files and directories in your present working directory.

#List 'all' the files in 'long' form with 'human' readable file sizes.

ls -alh

This is used to see the permissions of files/directories, the username and group of the file/directory owner, the file/directory size in bytes and the time of it's last modification.

**Kill and Killall**

Kill is used for killing a process using the process ID. Killall is used to kill *all* the processes started by the same program. Killall uses the process name.

#Kill process with the ID 436

kill 436

#Kill all processes started by the chrome program

killall chrome

By default kill allows the process to stop what it's doing and wrap things up before is stops. If you want to 'pull the plug' on a program and kill it immediately, use the -9 option.

# kill process with the id 567 immediately

kill -9 567

**Apt-get and Apt**

Apt-get is the standard command to install packages on all Debian based systems. apt is a shorthand version that works the same way. If you want the package to be installed without out further questioning from the system, you can use the -y flag

# Install the nano package

sudo apt-get install nano

# Install the top package without asking for confirmation

sudo apt-get -y install top

**Nano and text editors**

Nano is a basic text editor in Linux. There are other text editors listed here as well for you to try. If you decide to try these, google them first to learn about how they work. The man pages are also a good resource.

# open my\_doc.txt with the nano text editor

nano my\_doc.txt

# open my\_doc.txt with the gedit text editor

gedit my\_doc.txt

# open my\_doc.txt with the vi text editor

vi my\_doc.txt

# open my\_doc.txt with the emacs text editor

emacs my\_doc.txt

**sudo**

Stands for 'Super User Do'. It's the command you have to use if you want to invoke the system permissions of the root user (also known as the super user).

# Show the contents of the /etc/shadow file

sudo cat /etc/shadow

# Update the list of programs offered in the `apt` repository

sudo apt update

Many files/directories are only accessible by the root user. Also many programs require root permissions to run. If you are not logged in as root, you either have to switch your login to the root user, or you can use sudo command to run a single command with root permissions. Note: In order to use the 'sudo' command, your user has to be part of the 'sudo' group.

You can see what commands are available for your user with the -l flag

# Print the available sudo commands for the current user

sudo -l

If you want to see what commands are available for another user, add the -U flag and the username

# Print the available sudo commands for the user mike

sudo -lU mike

**su**

Stands for 'Switch User'. If you do not specify a user to switch to, the default is root.

# Switch to user mike

su mike

# Switch your login to the root user and 'preserve' your current environment.

su -p

**visudo**

You *must* use visudo to edit the /etc/sudoers file.

# Edit the /etc/sudoers file and validate that it is not damaged before saving.

sudo visudo -c

**chage**

Chage allows an administrator to set expirations on passwords, along with setting how many days before the next password change.

To see all of the chage info for a user, use the -l flag

# look at the chage info for the user mike

sudo chage -l mike

To set the Maximum number of days between password changes use the -M flag

# set the password to expire after 90 days for the user mike

chage -M 90 mike

To set the password to expire immediately, use the -d flag with the value 0

# set the password to expire immediately for the user mike

chage -d 0 mike

**id**

The id command gives you the UID, GID, and group information for a user.

#Show the UID, GID, and group information for user randal.

id randal

If no user is specified, the current user's info is displayed.

**adduser**

adduser makes it easy to add a user with their password and user info.

# Add a new user with the username 'ralph'

sudo adduser ralph

If you want to create a system user use the --system option to give the user a UID < 1000. Use the --no-create-home option to avoid creating a home folder.

# Create a system user named http without creating a home folder

sudo adduser --system --no-create-home http

**addgroup**

addgroup allows you to make and create groups on the system.

# Create a new group named 'developers'

addgroup developers

**usermod**

usermod allows you to change many parameters of a user. It is typically used to change a user's primary group, or add/remove secondary groups.

# Add the user bertha to the group hr\_administrators

usermod -aG hr\_administrates bertha

# Remove the user jack from all groups except the jack group

sudo usermod -G jack jack

**deluser**

deluser allows you to easily delete a user from the system. If you would like to also remove their home folder and files, use the --remove-home flag

# remove the user torbin from the system and delete his home folder

sudo deluser --remove-home torbin

**delgroup**

delgroup lets you easily remove a group from the system

# remove the slackers group from the system

sudo delgroup slackers

**chmod**

You change permissions with the command chmod, which stands for "change mode".

# Change the permissions to rwx for the user, rw- for the group and --- for everyone else.

chmod u+rwx,g=rw,o= permissions\_file

You can also use chmod with octal notation to set permissons.

# Change the permissions to rwx for the user, rw- for the group and --- for everyone else.

chmod 760 permissions\_file

**chown**

Stands for 'change owner'. When you change the owner, you also have to specify the group that you are assigning.

# Change the owner to bernard and the group to finance for the file spreadsheet

chown bernard:finance spreadsheet

Anything else about the Command

**Passwd**

The passwd command lets you change the password for a user.

#change the password for the user maxwell

sudo passwd maxwell

**Find**

Find is a great search tool to search for any files or folders on the system. To speicify between files or folders, user the -type flag along with f for files or d for directories.

# find all \_files\_ in the /etc directory

find /etc -type f

# find all \_directories\_ in the /etc directory

find /etc -type d

If you would like to search for a string in the name of the file, use the -iname flag along with the -type flag

# find a file with 'shadow' in the name inside the /etc directory

find /etc -iname shadow -type f

To search for files with particular permissions, use the -perm flag

# search for files that have the 2000 (SGID) bit set inside /usr/bin

find /usr/bin -perm /2000 -type f

# search for files that have the 4000 (SUID) bit set inside /usr/bin

find /usr/bin -perm /4000 -type f

**Systemctl**

systemctl allows you to start, stop, enable, disable and get the status of a service, as well as view all the services installed and running on the system. To get the status of a service use status and to see all the services, use -t for type along with service and --all to get everything on the system.

# search for \_all\_ services that are currently running on the system

systemctl -t service --all

# get the status of the apache2 service

sudo systemctl status apache2

If you want to start or stop a service, simply use start or stop flags

# start the apache2 service

sudo systemctl start apache2

# stop the apache2 service from running

sudo systemctl stop apache2

systemctl allows you to enable and disable a service from starting automatically when the system boots up. To set either of these, simply use enable or disable

# stop the apache2 service from starting automatically when the system starts up

sudo systemctl disable apache2

# set the apache2 service to start automatically when the system starts up

sudo systemctl enable apache2