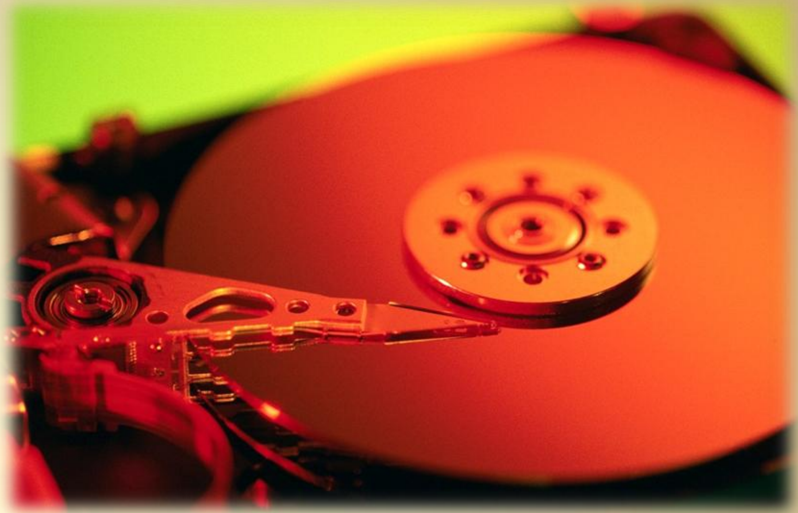


Backups

This material is based on work supported by the National Science Foundation under Grant No. 0802551

C3L7S1

Lesson Overview

You are a Linux administrator for a local hospital. One morning, as you report for work, you are met in the lobby by a panicking Director of Patient Records. He tells you that their database server crashed last night, and they are not able to access out-patient records for today's appointments.

Your first question is, "what server?" since you did not know that Patient Records had a separate server. Your second question is, "when was the last backup done?" but instead of an answer, you get a blank stare as the director asks, "what's a backup?"

In this lesson, you will be introduced to the backup process for Linux systems and helpful tools that will automate this critical task.

It is important to learn and apply good backup procedures to restore and safeguard data in case of system failures, data loss, data corruption, malicious intrusions, accidental erasures, and legal requirements to preserve data.

Select **PLAY** below to review the lesson introduction:

View Video VideoLesson7BackupsIntroduction(C3L7S2).mp4

Objective

You should know what will be expected of you when you complete this lesson. These expectations are presented as objectives. Objectives are short statements of expectations that tell you what you must be able to do, perform, learn, or adjust after reviewing the lesson.

Lesson Objective:

Given the need to back up data locally or over a network, the student will be able to evaluate popular backup technologies, select an appropriate backup tool, and configure it for regular backup purposes using a Linux BASH or PHP script per industry standards.

Lesson Outline

In this lesson, you will explore:

- Backups
  - Need for backups
  - Backup Methods
- Tar Utility
  - Using Tar for backups
  - CPIO
- Rsync Utility
  - Using Rsync
  - Portable backups
  - Remote backups

Backups

A backup is a process in which existing data is copied and stored. If the original data is erased or corrupted, the copied data may be used to restore the source back to its copied state.

Backups are a safety net. They protect the data, the applications, and the configurations from loss due to hardware failure, software changes, and natural disasters.

A good backup should copy all required files to another media, such as an external hard-drive, a tape drive, another server, CD-ROM, or DVD. In an ideal scenario, the copied data should be stored at another physical location or taken off-site. Having a backup stored in the same area of the server will not do much good if the location is flooded, burnt, or if the equipment is stolen.

For many companies, the data on their servers is their biggest asset and the loss of this data could put the company out of business. Consequently, having a valid backup plan is not only good business, it is essential!

Planning for Backups

The backup plan is designed for each physical machine that needs to be backed up. It identifies what is backed up, when it is backed up, and to what media it will be copied.

The backup plan, in many cases, also identifies where the backup is stored once it is completed. All organizations also need to decide how long the backups are stored, when they are destroyed, and in what manner they are destroyed. This processes are very important for healthcare organizations that are required to adhere to Health Insurance Privacy Act (HIPA) and other regulations.

Lost or stolen backups can lead to identity theft, security leaks, privacy violations, and possibly theft of intellectual property.

It is important to remember that backups are done using the root security level and most backups will likely copy all data to the backup media.

Tape Backup Plan

While there are several available backup plans, the most frequently used have not changed since the early days of Unix. In the early days of computing, backups were originally done using tape drives, and each backup was called a backup set. Review the backup schedule below:

Backup Sets	Description
Set 1	Sunday Morning – Full Backup (Everything on the machine)
Set 2	Monday Morning – Only the changed data since Set 1
Set 3	Tuesday Morning – Only the changed data since Set 2
Set 4	Wednesday Morning – Only the changed data since Set 3
Set 5	Thursday Morning – Only the changed data since Set 4
Set 6	Friday Morning – Only the chanced data since Set 5
Set 7	Saturday Morning – Only the changed data since Set 6
Set 8	Sunday Morning – Full backup (Everything on machine)
Set 9	Monday Morning – Only the changed data since Set 1

[Backup plan continued on next slide . . .]

Tape Backup Plan (Contd)

[Continued from previous . . .]

Backup Sets	Description
Set 10	Tuesday Morning – Only the changed data since Set 9
Set 11	Wednesday Morning – Only the changed data since Set 10
Set 12	Thursday Morning – Only the changed data since Set 11
Set 13	Friday Morning – Only the chanced data since Set 12
Set 14	Saturday Morning – Only the changed data since Set 13
Set 15	Sunday Morning – Full backup (Everything on machine)
Set 16	Monday Morning – Only the changed data since Set 15
Set 1	Tuesday Morning – Only the changed data since Set 15
Set 2	Wednesday Morning – Only the changed data since Set 1
Set 3	Thursday Morning – Only the changed data since Set 2
Set 4	Friday Morning – Only the chanced data since Set 3
Set 5	Saturday Morning – Only the changed data since Set 4
Set 6	Sunday Morning – Full backup (Everything on machine)

Tape Backup Plan Explained

By using this backup schedule, the system administrator always has two full weeks of backups. If a restore is needed, the system administrator can start with the Sunday data set that has a full backup and then restore the incremental changes up to the point at which data was lost.

If the data was lost mid-day, an administrator would have to do some extra work to restore the data back to the point just before the failure.

Looking at the backup plan, you will notice there are 16 sets of tapes, and the tapes are re-used every 16 days. This re-use destroys the old backups by over-writing them with new data.

Most organizations also created a year-end backup after the close of business at the end of their fiscal year. This yearly backup is stored for a period of time based on the data retention requirements of the Internal Revenue Service.

Protect Backups

The storage location for backups is very important. In times past, there were fewer options for backup storage. The tape operator ran the backup, the tape was removed from the tape drive and handed off to someone to put in a vault on-site, or better yet, removed from the location and stored in a vault off-site.

For many small businesses, the owner would be responsible for taking the backup to his/her home for safekeeping. In some cases, armored car services will pick up backups daily and deliver them to a central storage facility, such as Iron Mountain. However, as technology changed, tapes and physical media are no longer the primary form of available backups.

After Tape Backups

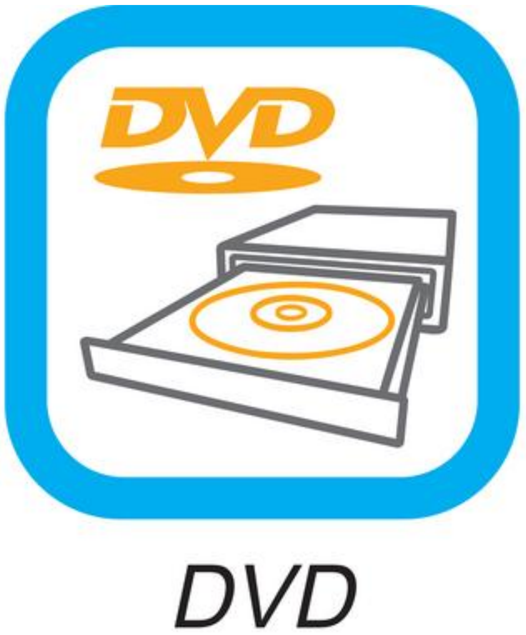
Over the last ten years, the hard drive space became very inexpensive and businesses began storing more data on these drives. In the early 1980's, the average hard drive was less than 20 Megabytes and cost about $300.00. Today, drives with 1 Terabyte of storage are available at local computer supply stores for under $100.00.

Many laptops and desktops today are sold with drives larger and faster than tape drives. Tape backups are slow because data must be accessed sequentially (in order from start to finish). If a file has changed since the last backup, the single file cannot be over-written on the tape. Instead, you must over-write the whole tape which is a time-consuming process. Consequently, administrators began looking at other media for backups.

CD & DVD Backups

One of the first and most frequently used forms of storage media is the CD-ROM. For many businesses with laptops or small systems, it was enough for an administrator to sit and copy the data to the CD-ROM once a month and store it forever. But CD-ROMs are read-only and only hold 700 MB of data. For those with 500 Gigabyte drives in their laptops, it takes a large number of CDs even if the data is compressed.

The next form of backup was the DVD, which can hold almost 5 GB of data on a single side. Again this storage medium worked for a time and may still work for incremental backups, but a full backup may require more than 100 expensive DVDs. Additionally, backups spanned several DVDs and each had to be switched manually unless businesses owned an expensive DVD changer to automate the process. A better solution was needed.



External Drives & Servers

The current method of backup for many is to copy their the data to large external hard drives, servers, or data centers. Many companies with a mobile workforce, work-at-home users, or a large infrastructure maintain a data-center with one or more servers specifically designed as backup servers.

On a nightly basis, every machine connects to this server and backs up it's data. Some companies will then backup this server to an second server so that there is not a single point of failure possible.

Differential Backups

Differential backups employ a copying process where only changed data is backed up. Each backup only transfers the data that has changed since the last time that that “set” was used. This method is also known as a differential backup, meaning we are backing up anything that has changed, not the entire system.

Differential backups are possible because unlike tapes, which are sequential, disk space is random access. This means that if a file in the middle of the backup has changed, it can be updated without having to re-write the entire backup. In other words, differential backup synchronizes the data between the machine being backup up and the backup server or hard drive.

In Linux, there are several tools available to do this synchronization.

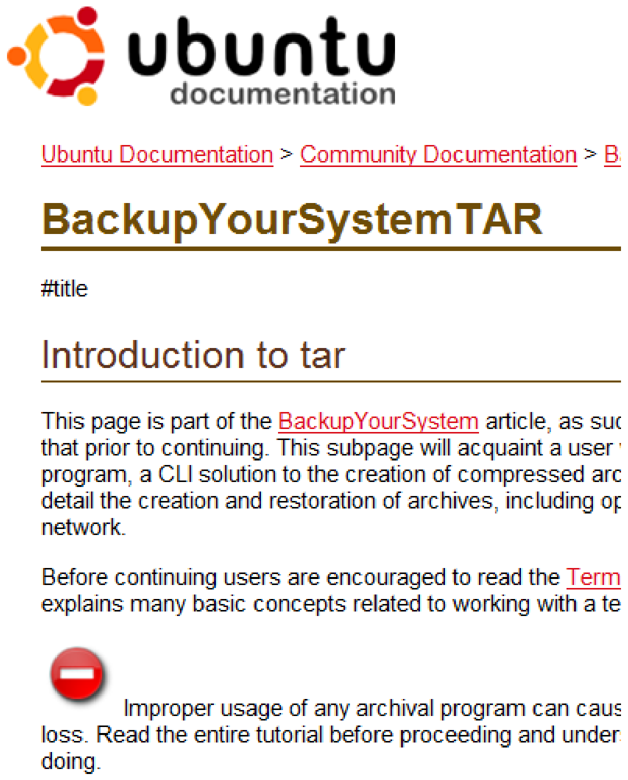
Linux Tools for Backups

The three backup tools we will explore in this lesson are TAR, CPIO, and Rsync. Each of these tools has a specific purpose, and each has some limitations.

TAR was the first such storage tool to become available. TAR is an acronym for "tape archive," and it creates and manipulates streaming archive files. TAR will also start with the directory specified and work recursively through each sub-directory. This means that it will work its way down the file tree, looking into each folder within the folder that was specified and copying all its contents unless told otherwise.

For some backups, this process was a limitation since it was impossible to only copy specific portions of a directory such as when you wanted to backup only some of the configuration files in the */etc* directory.

Complete example 1 on the next slide as a demonstration of TAR as a backup method.



ubuntu documentation

Ubuntu Documentation > Community Documentation > B

BackupYourSystemTAR

#title

Introduction to tar

This page is part of the BackupYourSystem article, as suc
that prior to continuing. This subpage will acquaint a user
program, a CLI solution to the creation of compressed arc
detail the creation and restoration of archives, including op
network.

Before continuing users are encouraged to read the Term
explains many basic concepts related to working with a te

Improper usage of any archival program can caus
loss. Read the entire tutorial before proceeding and under
doing.

Example 1 Tar Backups

In your Linux installation, open a terminal window and on the command line enter the following and press [Enter].

```
tar cvf sample-etc.tar /etc
```

Once the system returns you to a command prompt, enter the following and press [Enter].

```
ls –l sample-etc.tar
```

You will see a rather large file that contains the contents of your */etc* directory. This is one of the limitations of Tar. When used on its own, Tar creates a file that is the same size as the directory it is copying, and you are unable to modify a file within that directory. As Tar evolved, compression technology was added to the program, and external compression utilities, such as *Gzip* and *Bzip2*, could be used with Tar.

Tar has one important command line option that will decrease the workload of Linux administrators. If you put a **-t** on the command line as an option for Tar, it will test the action but not perform (execute) it. This option is extremely important to an administrator logged in as *root* or *sudo*. When using Tar, it is best to test your options first!

Required Reading:

- GNU Tar Manual
- Backup with Tar
- Tar on Ubuntu

Example 2: Compressed Tar Backups

In your Linux installation, open a terminal window and on the command line enter the command below and press [Enter]:

```
tar cvfz sample-etc.tar.gz /etc
```

Once the system returns you to a command prompt type the command below and press [Enter]:

```
ls –l sample-etc.tar*
```

You should now see your tar file from Example I and a new *tar.gz* file. Look at the differences in file sizes between the two files. The *sample-etc.tar.gz* file should be much smaller than the original file. But copying the smaller file to the server will still take time, and new copies have to be made each time the files are modified.

Required Reading:

- Tar man page

Example 3: Bzip2 and Tar

Bzip2 is an updated compression technology available for Tar. In your Linux installation, open a terminal window and enter the following command on the command line and press [Enter].

```
tar cvfj sample-etc.tar.bz2 /etc
```

Once the system returns you to a command prompt, enter the command below and press [Enter]:

```
ls –l sample-etc.tar*
```

You should now see your tar file from Example 1, Example2 and a new *tar.bz2* file. Notice the differences in file sizes. The *sample-etc.tar.bz2* file should be much smaller than the original file, and slightly smaller than the file generated in Example 2—the .gz file.

bzip2

bzip2 and libbzip2

What is bzip2?

bzip2 is a freely available, patent free (see below)
statistical compressors), whilst being around twice

The current version is **1.0.6**, released 20 Sept 20

Version 1.0.6 removes a potential security vulnera

Why would I want to use it?

- Because it compresses well. So it packs more s
  distance network traffic, etc. It's not the world's
- Because it's open-source (BSD-style license), a
  Caveat emptor). So you can use it for whatever
- Because it supports (limited) recovery from me
  be able to decompress those parts of the file w
- Because you already know how to use it. bzip2'

From: http://bzip.org

Backup: Specific Files Only

The backups we have explored so far are files that can be copied to a backup location (CD-ROM, DVD, External Hard drive, or server) and kept for a period of time. But the backup will include everything in the directory and cannot be modified.

As the size of the backup systems grow, you will find it necessary to select only certain files for backup purposes. Certain files that rarely change or are not critical may not need backup, while other files may need to be preserved every time they change.

CPIO is a utility that helps a user or administrator backup up specific files when necessary.

Required Reading:

- CPIO Quickstart

Selective Backup with CPIO

For Linux servers, some of the most important files for backup are configuration files. It is possible, and sometimes desirable, to do a clean system install from the Linux Distribution installation media in the event of a system failure but recreating the configuration files, users, and passwords, may take days, if not months. Consequently, configuration files are usually a target for frequent backups.

Many system administrators will identify and backup certain files on a regular basis by burning them to CDs or writing them to protected directories on a second server for this reason.

CPIO allows the administrator to specify a list of files that should be copied to the backup file. CPIO also interfaces with Tar so that only the files specified are added to the Tar file. One way to specify the files is to use a text file that contains the full paths of the files you wish to copy. Review Example 4 for a demonstration.

Required Reading:

- CPIO Quickstart

Example 4: Using CPIO

Instructions	Command Line Input
Open your Linux Terminal	
Type	mkdir –p ~/code/Lesson7
Access directory **lesson7**	cd ~/code/Lesson7
Type	vi Example4.data
Input the following:	/etc/passwd /etc/group /etc/shadow /etc/samba/smbpasswd /etc/ssh/sshd_config /etc/network/interfaces
Save and exit the file in command mode	:x
Type	sudo cat Example4.dat |sudo cpio -o -H tar > Example4.tar
Enter password if asked by sudo	
Execute script with non-existent file	ls –l

Example 4: Explained

Example 4 will create a tar file containing only the files specified on the command line. This file is not compressed or encrypted and can restore a copy of the files by entering the command below at the command prompt.

tar xvf ~/code/Lesson7/Example4.tar

Tar will remove the leading "/" from all paths by default and create the directory structure under your current directory.

Give it a try and then type **ls –l** to see the result.

If you want to see what is inside the archive without performing an actual restore type:

tar tvf ~/code/Lesson7/Example4.tar

While Tar is one way to restore files, you may also use CPIO. Let's try another example to see how this works.

Required Reading:
- CPIO Tutorial

Example 5: Using CPIO with Tar

Instructions	Command Line Input
Open your Linux Terminal	
Access directory **lesson7**	cd ~/code/Lesson7
Verify you are in the lesson 7 directory. If not, access lesson7 directory	pwd
Type	rm –rf etc
Type	tar xf ~/code/Lesson7/Example4.tar | cpio -i
Save and exit the file in command mode	ls –l etc

Ignore any error messages from CPIO. The **ls** command in step 6 should reveal all the files you listed for backup in your *Example4.dat* file from Example 4.

Creating a file list is a good option for a short list of files but this process is time consuming if the list is long. Fortunately, you can use the find command to search for a file pattern. Let's try this in example 6.

Example 6: Using Find and CPIO

Instructions	Command Line Input
Open your Linux Terminal	
Access directory **lesson7**	cd ~/code/Lesson7
Verify you are in the lesson 7 directory. If not, access lesson7 directory	pwd
Type	sudo find /etc -iname "*config
The command above finds all files under the /etc directory that contains the string "config."	
Enter your password if required by sudo	
Type	sudo find /etc –iname "*config"| cpio –o –H tar > Example6.tar
Type	tar tvf Example6.tar
Save and exit the file in command mode	ls –l etc

Example 6: Explained

```
sudo find /etc –iname “*config”| cpio –o –H tar > Example6.tar
```

The command above copies all files with the string “config” to the archive file Example6.tar, which could be copied to a backup location.

In the event of a system failure, the Example6.tar file could be used to restore all the config files within the archive.

```
tar tvf Example6.tar
```

The code above displays all the files that were copied to the backup Tar file (Example6.tar). Unfortunately, Tar files are not encrypted or compressed by default. Tar files are virtual directories that contain archived files within them.

Try opening *Example6.tar* with Vi or any other text editor. You should be able to read the contents.

Required Reading:
- Tar command
- CPIO Under Linux
- GNU CPIO

Review of Tar & CPIO

As mentioned earlier in this lesson, when using Tar and CPIO, you are creating a sequential file. This sequential file does not allow you to change the contents without re-writing the entire file.

For discussion sake, let's say you used CPIO to create a tar file containing your entire /etc directory. Then you added a user. In order to capture the new password and shadow file, you would need to create an entire new backup file, not just the changed files.

In the event you are using read only media, such as a CD or DVD, this could get expensive. In the event you are using a tape drive, this can be time consuming for a large system. Fortunately, Linux brought us a command line application that allows us to "update" a backup and copy only the changed files to a backup device as long as that device is a random access device and read-write capable.

Required Reading:
- Tar & CPIO
- CPIO

Rsync

Backups

Introduction to Rsync

With the advent of large hard drives, backup servers, and off-site backups in data centers, it became possible to do non-sequential (or random access) backups. This meant that only changed files would need to be copied to the server during a full system backup.

The large hard drives meant that more data could be stored and more full backups could be kept. The primary application used for these backups is Rsync.

Required Reading:
- Rsync

How Does Rsync Work?

Rsync is designed to speed the backup process by copying the differences between an old file (or file system) and the new version of the file system (or file systems) rather than the entire file system. Additionally, these small differences in file sizes are compressed before they are copied. This capability becomes valuable when you are copying a complete file system or large directories for daily backups.

Rsync has the ability to synchronize a file system to a local directory, a second hard drive on a local machine, an external hard drive, a network hard drive, or a remote server.

Rsync is available for Linux, any other BSD or Unix based operating system such as Apple's OS/X, as well as Microsoft Windows. This cross platform availability means that backup scripts using Rsync can be executed on most production platforms.

In the next example, you will run a basic Rsync command.

Required Reading:
- Rsync
- Rsync Tutorial

Example 7: Basic Rsync Command

Follow the example below to explore the use syslog command:

Instructions	Command Line Input
Open terminal window	
Make directory	mkdir –p ~/code/Lesson7a
Access directory	cd ~/code/Lesson7a
Enter	sudo rsync –azv /etc ./
Enter	ls
Enter	ls –lR etc
Enter	diff -r /etc etc

Example 7: Description

In line 2 you created a directory to use for this example.

In line 3 you changed to that directory.

Line 4 contained the Rsync command that copies the /etc directory to the Lesson7a directory. The command option, **azv** tells Rsync to copy the files with all attributes, compress them as they are streamed, and to be verbose (show the activity) as the copy is being made.

Line 5 displays a single directory **etc.**

Line 6 shows a full list of all the files in the **etc** directory.

Line 7 is used to show if there are differences between the files in **/etc** and files in **~/code/Lesson7a/etc** directory. The command structure in line 7 can be used to verify that the backup completed properly.

Command Line Input
mkdir –p ~/code/Lesson7a
cd ~/code/Lesson7a
sudo rsync –azv /etc ./
ls
ls –lR etc
diff -r /etc etc

Rsync Syntax

The Rsync command line has the following syntax:

```
rsync [options] user@from-host:source-file user@to-host:destination-file
rsync [options] source-file destination-file
```

The first line includes user and host information for copying to and from remote systems. The second line is for a source file and destination files for copying within a single system or to removable drives.

Remember, in Linux a removable drive is mounted as a file path. The English translation of the Rysnc command is to "rsync from this location to that location."

If you wish to copy an entire directory, you may put a trailing slash (/) at the end of the source file name. The slash signifies that everything in the directory (source-file) will be copied recursively. If you do not add the trailing slash after the directory name, only the base directory name will be copied.

Required Reading:

- Rsync Examples
- Rsync Tutorial

Rsync Command Line Options

Options	Description
-a	Preserves the ownerships, permissions, and modified dates of the copied files
-v	Displays information about what rsync is doing
-z	Compresses files while copying them
-u	Skips files that are newer in the destination-file than the source-file
--delete	Deletes files in the destination-file that are not in the source-file. Be careful, this option can remove files that you did not intend to remove
--dry-run	Runs rsync without writing to disk. When combined with the –v option, this option reports on what rsync would have done had it been run without this option. Use with the --delete option to test what will happen prior to running it
--backup	Renames files that otherwise would be deleted or overwritten. By default, the rysnc renames files by appending a tilde (~) to existing filenames.
--backup-dir=**dir**	When used with the --backup option, moves the files that otherwise would be deleted or overwritten into the directory named **dir**. After moving the older version of the file in **dir**, rsync copies the newer version of the file from **source-file** to **destination-file.** The directory named **dir** is a located on the same system of **destination-file.** If **dir** is a relative pathname, it is relative to **destination-file**.
-x	Only copies file from within the same file system. It will not cross devices.

Rsync with SSH

The following command is a valid rsync command:

rsysnc –azv –e ssh testuser@testsystem1.com:* testuser1@testsystem2.com:

This command will copy the contents of the testuser's home directory on *testsystem1.com* to the testuser1 home directory on testsystem2.com.

The options (**–azv –e ssh)** indicate that Rsync will compress the files as they are sent, preserve all ownerships, permissions, and modified dates. The Rsync utility will show all activities as they are completed and will use ssh to encrypt the transfer so plain text is not being sent.

Rsync is a very efficient tool for copying data (files, directories, or entire file systems) to a second machine as well. You may also add the machine's IP address or fully qualified domain name into the command line and stream backups across the Internet. For Internet transmissions, you can use ssh encryption to provide secure transfers and backups. If ssh is setup correctly on both machines, un-attended backups can be properly completed.

Follow example 8 to setup a second user for an Internet backup on your virtual machine.

Required Reading:

- Rsync Expert

Example 8: Remote Backup

Use the Linux administration tool of your choice and create a second user on your machine (virtual or actual).

Log into this user account and make sure you can access a command line terminal. Remember this user's name and password. For the remainder of this lesson, you will substitute this name for any occurrence of **username2** in the directions.

Instructions	Commands
Make directory from terminal of username2	mkdir ~/code
Show contents to ensure code directory exists	ls
Log off user2, and log into the user account you used for previous examples. Open a terminal window.	
Go to code directory	cd ~/code
Secure Shell (Accept the SSH key if asked)	ssh "username2"@127.0.0.1
Enter password for user2, then list the contents	ls
Verify the code directory exists. If not, create it. Then exit.	Exit
Access code directory	cd ~/code

[Instructions continue on next slide . . .]

Example 8: Remote Backup (Contd)

[Continued from previous . . .]

Instructions	Commands
Type	rsync –azv –e ssh * “username2”@127.0.0.1:code/
Enter username and password for user2	ls
Type	cd ~/code
Type	ls –lR

Notice that the contents of your first user’s *code* directory is now in the second user’s code directory.

Example 8: Description

In example8, you created a new user ID on your local machine and used SSH to access it. The IP address of *127.0.0.1* always routes you back through TCP/IP to your own machine. Using this local IP is a great way to test network functions without a second machine.

Then, you created the *~/code* directory on the second machine. This step was unnecessary, but creating the directory protects against errors and ensures you can write to it.

Next, you tested the ssh connection and ensured you had the correct passwords. You also verified that the code directory your created existed. Then you performed the backup routine using this command: **rsync –azv –e ssh * "username2"@127.0.0.1:code/**

Finally, in step 18 you logged back into your "remote" machine using ssh and verified the backup of the code directory was completed properly. If the directories match, then your backup was successful.

But how does Rsync help if you lose files? Follow the directions in example 9 to practice restoring a backup from a remote location.

Example 9: Restoring Backup

Instructions	Commands
Open a terminal window using your main user id	
Access the code directory	cd ~/code
Type	rm –rf Lesson7a
Type	**ls**
Verify that subdirectory Lesson7a is deleted	
Restore using SSH and Rsync	rsync –azv –e ssh “username2”@127.0.0.1:code/* ./
Enter password for second user	
List contents	ls
Verify that directory Lesson7a has been restored	

Example 9: Description

The difference between backing up and restoring Rsync command is the order of the options.

Rsync uses a standard copy command structure:

Rsync -azv –e ssh source destination

The source and destination options can just be a directory, or can be a full user and machine name along with the directories to be copied.

There are additional command line options that can be used. One of the most frequent is the **--delete** option. This option deletes any file in the destination file structure that does not exist in the source file structure. It is a very useful to create a current mirror backup of the system.

Required Reading:

- Rsync Exclude Options

Rsync as a Server

While it is outside the scope of this lesson, Rsync can be run as a server using the **rsync** command. In server mode, Rsync waits for a network connection and allows access to pre-determined paths. These paths can be restricted on a user basis or defined in a configuration file.

Many Linux distribution mirror sites are created and maintained using Rsync. A central Rsync server and associated mirror systems connect on a regular basis to synchronize with the source distribution. Large distribution sites such as those handling Debian, Fedora, Ubuntu, and KDE updates use a tree structure to distribute changes.

Another use of the Rsync server is to create a local copy of software updates to decrease external network traffic from within an organization.

Required Reading:
- Rsync Tutorial

Rsync and Databases

Many backups will need to include databases. It is critical that before backing up the database, you dump the database files to a text file.

For Mysql databases, use the **mysqldump** command. For Postgresql databases, use the **pg_dumpall** command.

To restore one of the above databases, use Rsync to restore the dump file and then restore the dump file to the database.

Remember that relational databases may be in the middle of an update, save, or delete process whenever the system fails or data becomes corrupted. Therefore, it is best not to rely on Rsync tables by themselves to backup databases.

Required Reading:
- PosgreSQL backup
- MySql Dump man page

Using a Modern Backup Schedule

At the start of this lesson, we explored a backup routine that was based on maintaining 16 days of backups on sequential tapes. Every 16 days, the tape set was over-written with a new file dump.

There has been a change in the thought process on incremental backups with random access drives and larger hard drives.

Instead of using tape sets, most businesses use drives (perhaps USB drives, internal and external hard drives) or just directories on a remote server. The second change in backup strategy is to use more full backups rather than partial backups.

Random Access Backups

Days	Backup Schedule Description
Day 1	Sunday Morning – Full Backup #1(Everything on the machine)
Day 2	Monday Morning – Full Backup #2
Day 3	Tuesday Morning – Update Drive 2
Day 4	Wednesday Morning – Update Drive 2
Day 5	Thursday Morning – Update Drive 2
Day 6	Friday Morning – Update Drive 2
Day 7	Saturday Morning – Update Drive 2
Day 8	Sunday Morning – Full Backup #3 (Everything on the machine)
Day 9	Monday Morning – Full Backup #2
Day 10	Tuesday Morning – Update Drive 2
Day 11	Wednesday Morning – Update Drive 2
Day 12	Thursday Morning – Update Drive 2
Day 13	Friday Morning – Update Drive 2
Day 14	Saturday Morning – Update Drive 2
Day 15	Sunday Morning – Full Backup #1(Everything on the machine)
Day 16	Monday Morning – Full Backup #2

"Drive 2" may refer to a drive or a directory location on a remote server.

Scheduling Backups

Backups may be created manually by typing the Rsync command into the terminal window. However, the most effective way to begin backups is to configure them as cron jobs to run during off-peak work hours.

Let’s do one more example to explore this type of backup.

Required Reading:

- Using Cron on Ubuntu

Example 10: Random Access Backup

Instructions	Command Line Input
Open a terminal window using your main user ID	
Type	cd
Type	vi backup.sh
Insert mode	I
Input code: Press [Escape] key to exit insert mode when finished.	#!/bin/sh rm –rf bu.2 mv bu.1 bu.2 mv bu.0 bu.1 rsync --archive --link-dest=../bu.1 memos/ bu.0
Save and exit the file	:x
Make the script executable	chmod u+x backup.sh
Type	./backup.sh

This script will create a new directory for each day (0-2) and then copy the user's memos directory into the new directory.

Once all directories are filled then the script will delete the oldest directory and start over again.

Lesson Summary

In this lesson, we began by discussing the importance of backups, the various backup schedules, and early industry standards for tape set backup rotation.

We continued the lesson by exploring the **Tar** command as used in backups. We discovered that Tar is sequential and captures everything in the directory path. Tar was originally used for a tape archive and is sequential, meaning you are unable to change parts of the file and must re-write the entire tar file to capture changes.

Next we examined CPIO—a selective copy utility that can create a tar file. Again, as with the tar command, once the backup file is created, CPIO does not allow you to alter individual parts of it. Cpio creates a sequential file as well, so the backup file cannot be changed. It has to be re-created to include updates and changes. One of the positive attributes of CPIO is that it will read from a configuration file or a stream to select the files and directories included in the backup.

We concluded the lesson by exploring Rsync—a synchronization tool that allows you to copy from directory to directory, directory to external file system, and from directory to external server over the Internet. One of the most important feature of Rsync process is that the utility maintains the file attributes and permissions. Additionally, unlike Tar and CPIO, Rsync is not sequential, meaning it is able to update the backup file system with changes without recreating the entire backup. It does this by using diff files, that is small files that only transmit the differences between the new and the old directory structure.