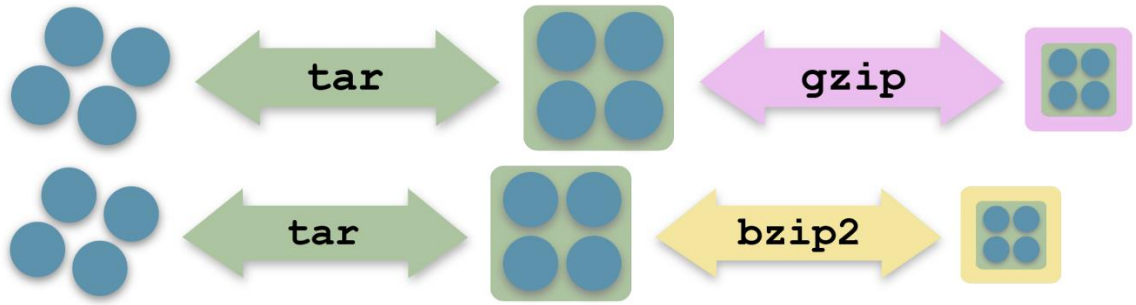
12.1 Introduction

Linux distributions provide several different sets of commands for compressing and archiving files and directories. This chapter will describe their advantages and disadvantages as well as their usefulness in making backup copies of files and directories efficiently.

More sophisticated types of copying, using the `dd` and `cpio` commands, will also be covered. These commands, while more complex, offer powerful features.



12.2 tar Command

An archive is a single file, which consists of many files, though not necessarily compressed. The `tar` command is typically used to make archives within Linux. These tar archive files, sometimes called tarballs, were originally used to backup data onto magnetic tape. Tar is derived from the words "**t**ape **ar**chive".

While the primary purpose of the `tar` command is to merge multiple files into a single file, it is capable of many different operations and there are numerous options. The functionality of the `tar` command can be broken down into three basic functions: creating, viewing and extracting archives.

The `tar` command accepts all three styles of options (`x`, `-x`, `--extract`) as parameters. Do not be surprised to see it used with options that have no hyphens (BSD style), a single hyphen (UNIX style) or two hyphens (GNU style).

The `tar` command originally did not support compression, but a newer version that was developed by the GNU project supports both `gzip` and `bzip2`compression. Both of these compression schemes will be presented later in this chapter. To use `gzip` compression with the `tar` command, use the `-z` option. To use `bzip2` compression, use the `-j` option. If a version of the `tar`command does not support `gzip` or `bzip2` compression, the `gzip` or `bzip2`commands can be used separately on the tar file.

To create a tar archive of the `/etc/systemd` directory, use the option `-c` to create and `-f` to specify a new file, followed by the directory to archive. Note that the `-f` option must be specified last, since it is indicating a filename:

```
tar -cf newfile.tar file
```

```
sysadmin@localhost:~$ tar -cf systemd-config.tar /etc/systemd
tar: Removing leading `/' from member names
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  systemd-config.tar
Documents  Music      Public    Videos     test
```

The `-v` verbose option will cause the `tar` command to display the files that are being included in the archive. If compression is going to be used, `gzip` for instance, then the `-z` needs to be added. File extensions are not relevant to Linux, but it is customary to add `.tar.gz` to the name of the compressed archive:

```
sysadmin@localhost:~$ tar -cvzf systemd-config.tar.gz /etc/systemd
tar: Removing leading `/' from member names
/etc/systemd/
/etc/systemd/system/
/etc/systemd/system/multi-user.target.wants/
/etc/systemd/system/multi-user.target.wants/rsyslog.service
/etc/systemd/system/multi-user.target.wants/bind9.service
/etc/systemd/system/multi-user.target.wants/ssh.service
/etc/systemd/system/syslog.service
/etc/systemd/system/sshd.service
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  systemd-config.tar     test
Documents  Music      Public    Videos     systemd-config.tar.gz
```

Use the `-t` option to the `tar` command to view a list (table of contents) of a tar file. Even if the archive file is compressed, the correct compression option does not need to be specified to view a tar archive file. Only the file `-f` and the list `-t`option are required to view the table of contents. Once again, note that the `-f`option is used last so that the filename can be specified as an argument to this option:

```
sysadmin@localhost:~$ tar -tf systemd-config.tar.gz
etc/systemd/
etc/systemd/system/
etc/systemd/system/multi-user.target.wants/
etc/systemd/system/multi-user.target.wants/rsyslog.service
etc/systemd/system/multi-user.target.wants/bind9.service
etc/systemd/system/multi-user.target.wants/ssh.service
etc/systemd/system/syslog.service
etc/systemd/system/sshd.service
```

To view the table of contents of the archive in a format similar to a long listing (like the `ls -l` command), add the verbose `-v` option:

```
sysadmin@localhost:~$ tar -vtf systemd-config.tar.gz
drwxr-xr-x root/root         0 2014-09-18 22:24 etc/systemd/
drwxr-xr-x root/root         0 2014-09-18 22:25 etc/systemd/system/
drwxr-xr-x root/root         0 2014-09-18 22:25 etc/systemd/system/multi-us
er.target.wants/
lrwxrwxrwx root/root         0 2014-09-17 03:37 etc/systemd/system/multi-us
er.target.wants/rsyslog.service -> /lib/systemd/system/rsyslog.service
lrwxrwxrwx root/root         0 2014-09-18 22:24 etc/systemd/system/multi-us
er.target.wants/bind9.service -> /lib/systemd/system/bind9.service
```

```
lrwxrwxrwx root/root         0 2014-09-18 22:25 etc/systemd/system/multi-us
er.target.wants/ssh.service -> /lib/systemd/system/ssh.service

lrwxrwxrwx root/root         0 2014-09-17 03:37 etc/systemd/system/syslog.s
ervice -> /lib/systemd/system/rsyslog.service

lrwxrwxrwx root/root         0 2014-09-18 22:25 etc/systemd/system/sshd.ser
vice -> /lib/systemd/system/ssh.service
```

To extract the files from the tar file, use the `-x` option. Normally, the `tar`command will attempt to extract the archive into the current directory, as demonstrated in the following example:

```
sysadmin@localhost:~$ tar -xf systemd-config.tar.gz

sysadmin@localhost:~$ ls -R etc

etc:

systemd


etc/systemd:

system


etc/systemd/system:

multi-user.target.wants  sshd.service  syslog.service


etc/systemd/system/multi-user.target.wants:

bind9.service  rsyslog.service  ssh.service
```

Use the `-C` option to specify an alternate directory to extract the contents into. This option is case sensitive and should not be confused with the create or `-c`option. The extraction process can also be made verbose by adding the `-v`option:

```
sysadmin@localhost:~$ tar -vxf systemd-config.tar.gz -C /tmp

etc/systemd/

etc/systemd/system/

etc/systemd/system/multi-user.target.wants/

etc/systemd/system/multi-user.target.wants/rsyslog.service

etc/systemd/system/multi-user.target.wants/bind9.service

etc/systemd/system/multi-user.target.wants/ssh.service

etc/systemd/system/syslog.service

etc/systemd/system/sshd.service
```

12.3 gzip and gunzip Commands

The `gzip` command is used to create a compressed archive file. Likewise, the `gunzip` command is used to view the contents of an archive file, as well as extract those contents.

The `gzip` command should be used with caution, since its default behavior is to replace the original file specified with a compressed version. In the following example, the `red.txt` file is replaced with the compressed `red.txt.gz` file after using gzip:

```
sysadmin@localhost:~/test$ ls red*
red.txt
sysadmin@localhost:~/test$ gzip red.txt
sysadmin@localhost:~/test$ ls red*
red.txt.gz
```

To avoid replacing the original version of a file when using `gzip`, use the `-c`option. This causes the `gzip` command to send the gzipped data to standard output, and given that the output of the `gzip` command is binary data it will need to be redirected into a file. Remember to capture the output of the `gzip`command and redirect it into a file use the > character:

```
sysadmin@localhost:~/test$ gzip -c numbers.txt > numbers.txt.gz
sysadmin@localhost:~/test$ ls numbers*
numbers.txt  numbers.txt.gz
```

Using the `gzip` command with the `-c` option and redirection created a gzipped file while leaving the original file intact. This can be useful as the gzipped file can be moved to an archive directory location while preserving the original file in its original location.

The `gunzip` command reverses what `gzip` does, so files will be uncompressed and the gzipped file will be replaced with the uncompressed file:

```
sysadmin@localhost:~/test$ gunzip red.txt
sysadmin@localhost:~/test$ ls red*
red.txt
```

To view the amount of compression of an existing archived file, use the `-l` option with `gunzip`:

```
sysadmin@localhost:~/test$ gunzip -l numbers.txt.gz
         compressed        uncompressed  ratio uncompressed_name
                 42                  10 -20.0% numbers.txt
```

While it supports recursion with the `-r` option, by default `gzip` attempts to replace the original file with the gzipped file. This leads to errors when the files being gzipped are not owned by the user that tries to `gzip` them.

```
sysadmin@localhost:~/test$ gzip -r /run
gzip: /run/network/.ifstate.lock.gz: Permission denied
gzip: /run/network/ifstate.gz: Permission denied
gzip: /run/resolvconf/interface/original.resolvconf.gz: Permission denied
gzip: /run/resolvconf/resolv.conf.gz: Permission denied
```

```
gzip: /run/utmp.gz: Permission denied
gzip: /run/rsyslogd.pid.gz: Permission denied
gzip: /run/crond.pid.gz: Permission denied
gzip: /run/crond.reboot: Permission denied
gzip: /run/sshd.pid.gz: Permission denied
gzip: /run/named/named.pid.gz: Permission denied
gzip: /run/named/session.key: Permission denied
gzip: /run/motd.dynamic.gz: Permission denied
```

In order to be able to recursively compress files with the `gzip` command, a user needs to have the correct permissions on the directories the files are in. Typically this is limited to directories within the user's own home directory.

For example, to use the `gzip` command recursively on the `~/example`directory, it would be successful in replacing regular files with a gzip archive files:

```
sysadmin@localhost:~/test$ mkdir ./example
sysadmin@localhost:~/test$ touch ./example/one ./example/two ./example/thre
e
sysadmin@localhost:~/test$ ls ./example/
one  three  two
sysadmin@localhost:~/test$ gzip -r ./example
sysadmin@localhost:~/test$ ls ./example/
one.gz  three.gz  two.gz
```

The `gunzip` command can also work recursively, assuming the user has the correct permissions. As it works, it removes the `.gz` extension from each file:

```
sysadmin@localhost:~/test$ gunzip -r ./example/
sysadmin@localhost:~/test$ ls ./example/
one  three  two
```

Consider This

Permissions can have an impact on file management commands, such as the `gzip` and `gunzip` commands. To `gzip` or `gunzip` a file within a directory, a user must have write and execute permission on a directory as well as read permission on the file. Regular users typically only have this type of permission in their home directory and its subdirectories.

12.4 bzip2 and bunzip2 Commands

The `bzip2` and `bunzip2` commands work in a nearly identical fashion to the `gzip` and `gunzip` commands. The differences are in the type of algorithm (how the files are compressed) used to compress the files and the `.bz2` extension is added or removed from the file name (rather than the `.gz` extension).

When a new compressed file is created from an existing file with the `bzip2`command, the `.bz2` extension is added to the file name. Using the `-v` option will cause `bzip2` to report the compression ratio after it has finished. The `gzip`command also supports the `-v` option, so a file can be compressed using both commands and the compression ratio compared to determine which command uses a better compression technique for that specific file.

```
sysadmin@localhost:~$ ls ./example/
one  three  two
sysadmin@localhost:~$ bzip2 -v ./example/*
  ./example/one:    no data compressed.
  ./example/three:  no data compressed.
  ./example/two:    no data compressed.
sysadmin@localhost:~$ ls ./example/
one.bz2  three.bz2  two.bz2
```

Note that there was no data to compress in the empty files, `one`, `two`, and `three`. Just like `gunzip`, the `bunzip2` command uncompresses the file and removes the `.bz2` extension:

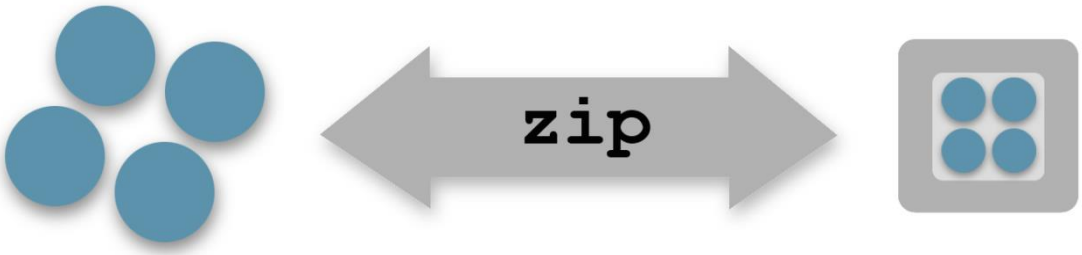
```
sysadmin@localhost:~$ bunzip2 -v ./example/*
  ./example/one.bz2:   done
  ./example/three.bz2: done
  ./example/two.bz2:   done
sysadmin@localhost:~$ ls ./example/
one  three  two
```

As demonstrated in the examples above, the `bzip2` command doesn't have an option `-r` to do recursion, so a wildcard can be used to match multiple files. The `bzip2` command also has an option `-c` to send the data to standard output, so it can be redirected to a new file:

```
sysadmin@localhost:~test$ bzip2 -c red.txt > red.txt.bz2
sysadmin@localhost:~test$ ls red*
red.txt  red.txt.bz2
```

12.5 zip and unzip Commands

Of the many commands available in Linux that take multiple files and combine them into one, the `zip` command might be most familiar. The functionality of zip compressed file management has been available in the personal computing world prior to Windows and is now included within the file management graphical utilities found within Microsoft's Windows and Apple's Mac OS X.



Although it is not a required topic to learn for the LPIC 1 exams, the `zip`command is very useful for creating archives that can easily be shared across multiple operating systems. The basic form of a `zip` command is:

```
zip [options…] destination files…
```

The destination file will automatically have a `.zip` extension added to it, if an extension is not specified. Also, the original files will not be replaced.

```
sysadmin@localhost:~$ zip ./example/package ./example/*
  adding: example/one (stored 0%)
  adding: example/three (stored 0%)
  adding: example/two (stored 0%)
sysadmin@localhost:~$ ls ./example/
one  package.zip  three  two
```

One especially useful option for the `zip` command is the `-r` option, which allows the `zip` command to recursively compress multiple directories into a single file. For example, to back up the files stored in the `/var/log/cups`directory, execute the following command:

```
sysadmin@localhost:~$ zip -r ./example/logfiles /var/log/cups/
  adding: var/log/cups/ (stored 0%)
  adding: var/log/cups/access_log.3.gz (stored 0%)
  adding: var/log/cups/access_log.2.gz (stored 0%)
  adding: var/log/cups/access_log.6.gz (stored 0%)
  adding: var/log/cups/error_log (stored 0%)
  adding: var/log/cups/error_log.1.gz (stored 0%)
  adding: var/log/cups/access_log.5.gz (stored 0%)
  adding: var/log/cups/error_log.3.gz (stored 0%)
  adding: var/log/cups/error_log.2.gz (stored 0%)
  adding: var/log/cups/access_log.1.gz (stored 0%)
  adding: var/log/cups/access_log (deflated 85%)
  adding: var/log/cups/access_log.4.gz (stored 0%)
  adding: var/log/cups/access_log.7.gz (stored 0%)
  adding: var/log/cups/page_log (stored 0%)
sysadmin@localhost:~$ ls ./example/
logfiles.zip  one  package.zip  three  two
```

Note that the log files had been gzipped prior to the use of the zip command.

The `unzip` command is used to extract the files from the zip archive file. Use the `unzip` command without options to extract a zip archive:

```
sysadmin@localhost:~$ cd ./example/
sysadmin@localhost:~/example$ unzip ./logfiles.zip
Archive:  ./logfiles.zip
   creating: var/log/cups/
 extracting: var/log/cups/access_log.3.gz
 extracting: var/log/cups/access_log.2.gz
 extracting: var/log/cups/access_log.6.gz
 extracting: var/log/cups/error_log
 extracting: var/log/cups/error_log.1.gz
 extracting: var/log/cups/access_log.5.gz
 extracting: var/log/cups/error_log.3.gz
 extracting: var/log/cups/error_log.2.gz
 extracting: var/log/cups/access_log.1.gz
  inflating: var/log/cups/access_log
 extracting: var/log/cups/access_log.4.gz
 extracting: var/log/cups/access_log.7.gz
 extracting: var/log/cups/page_log
sysadmin@localhost:~/example$ ls ~/example/var/log/cups/
access_log  access_log.1.gz  access_log.2.gz  access_log.3.gz  access_log.4
.gz  access_log.5.gz  access_log.6.gz  access_log.7.gz  error_log  error_lo
g.1.gz  error_log.2.gz  error_log.3.gz  sysadmin@localhost ~/example $
```

A new directory tree is created `~/example/var/log/cups/` that contains the files that were contained inside of the zip archive file.

To view the contents of a zip file without unpacking it, use `unzip` to view the list of its files with the `-l` option:

```
sysadmin@localhost:~/example$ unzip -l ./package.zip
Archive:  ./package.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2014-07-17 19:14   example/one
        0  2014-07-17 19:14   example/three
        0  2014-07-17 19:14   example/two
---------                     -------
        0                     3 files
```

12.6 xz Command

Another archival tool covered in the LPIC-1 objectives is `xz`. Using the `-z`option, `xz` can be used to compress a group of files individually.

```
sysadmin@localhost:~/test$ ls
adjectives.txt           alpha.txt    linux.txt     people.csv
alpha-first.txt          animals.txt  longfile.txt  profile.txt
```

```
alpha-first.txt.original  food.txt     newhome.txt   red.txt
alpha-second.txt          hidden.txt   numbers.txt
alpha-third.txt           letters.txt  os.csv
sysadmin@localhost:~/test$ xz -z *
sysadmin@localhost:~/test$ ls
adjectives.txt.xz            alpha.txt.xz    linux.txt.xz     people.csv.xz
alpha-first.txt.original.xz  animals.txt.xz  longfile.txt.xz  profile.txt.x
z
alpha-first.txt.xz           food.txt.xz     newhome.txt.xz   red.txt.xz
alpha-second.txt.xz          hidden.txt.xz   numbers.txt.xz
alpha-third.txt.xz           letters.txt.xz  os.csv.xz
```

The `-d` option can be used to uncompress the files just as easily.

```
sysadmin@localhost:~/test$ xz -d *
sysadmin@localhost:~/test$ ls
adjectives.txt            alpha.txt    linux.txt     people.csv
alpha-first.txt           animals.txt  longfile.txt  profile.txt
alpha-first.txt.original  food.txt     newhome.txt   red.txt
alpha-second.txt          hidden.txt   numbers.txt
alpha-third.txt           letters.txt  os.csv
```

However, it is often preferable to bundle to files before zipping them in any format, whether `bzip`, `gzip`, or `xz`. The following example archives the test directory using `tar` before compressing it with `xz`:

```
sysadmin@localhost:~/test$ cd ..
sysadmin@localhost:~$ tar -cf testdirectory.tar ./test/
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  test
Documents  Music      Public    Videos     testdirectory.tar
sysadmin@localhost:~$ xz -z testdirectory.tar
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  test
Documents  Music      Public    Videos     testdirectory.tar.xz
```

The `tar` command is also able to compress using `xz` directly using the `-J`option:

```
sysadmin@localhost:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
test
sysadmin@localhost:~$ tar -cJf testdirectory.tar.xz ./test/
sysadmin@localhost:~$ ls
Desktop    Downloads  Pictures  Templates  test
Documents  Music      Public    Videos     testdirectory.tar.xz
```

There are a huge number of options for the `xz` command, some relating to the compression ratio. Keep in mind when using `xz` that the more aggressive the compression, the harder the processor will have to work.

12.7 cpio Command

Another type of archive command that can merge many files into a single file is the `cpio` command. This command gets its name from two of its modes: copy-in mode and copy-out mode.

In copy-out mode, the `cpio` command will copy files from a directory into an archive. In copy-in mode, the `cpio` command will either list the archive file contents or copy files from an archive into a directory. It is easy to reverse these statements and get confused. Just remember that the archive is *outside* of the operating system.

There is a third mode called the copy-pass mode. In this mode, the `cpio`command copies files from one directory to another, which combines the copy-out and copy-in modes without creating an archive file.

To create a new archive file, the `cpio` command will run in copy-out mode taking a list of files from *standard input* and producing a file stream that can be redirected into a new archive file. Standard input in this case refers to the keyboard input by default, but this input could also come from the output of other commands.

The `-o` option puts the `cpio` command into copy-out mode. Using the `-v`option will cause the `cpio` command to list the files that it processes. So, to archive the current directory, execute the `ls` command and then send a list of the files into the `cpio` command as input by using the pipe `|` character (recall that the `>` character will capture the output of a command and put it into a file):

```
sysadmin@localhost:~$ ls | cpio -ov > archive.cpio
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
example
test
1 block
```

Consider This

Taking the output of one command and sending it into another command is called *redirection*. While redirection is discussed in detail in a later chapter, the following provides a brief introduction to the topic:

Consider the following pseudo command line: `cmd1 | cmd2`

The output of `cmd1`, which would normally be displayed in the terminal, is instead sent as standard input to `cmd2`. So, instead of having a user type data from the keyboard, `cmd2` uses the data from the output of `cmd1`.

The `find` command is a good way to generate the list of files to be sent to `cpio`.
The `find` command is automatically recursive, so it can be used to create a list of all files starting at a particular directory. For example, to archive the home directory and all of its subdirectories and files, execute the following command:

```
sysadmin@localhost:~$ find ~
/home/sysadmin
/home/sysadmin/.bash_logout
/home/sysadmin/.bashrc
/home/sysadmin/.profile
/home/sysadmin/.selected_editor
/home/sysadmin/Desktop
/home/sysadmin/Documents
/home/sysadmin/Downloads
/home/sysadmin/Music
/home/sysadmin/Pictures
/home/sysadmin/Public
/home/sysadmin/Templates
/home/sysadmin/Videos
/home/sysadmin/test
...
```

The `-v` verbose option is used to show the activity of the cpio command in the terminal.

To extract the files that are in a `cpio` archive, use the `-i` option with the `cpio`command to specify copy-in mode.. By default, `cpio` will not overwrite existing files unless the `-u` option is used. The `cpio` command will not create directories unless the `-d` option is used.

The `cpio` command also makes use of standard input to determine the name of the file that will be extracted from the archive. Therefore to extract files and directories, as well as overwriting existing files, execute the following:

```
sysadmin@localhost:~$ echo "/tmp/home.cpio" | cpio -iud
```

To specify the pass-through mode of the `cpio` command, specify the `-p` option. Again, if any directories are included, the `-d` option needs to be specified. To copy all the files from the home directory to a directory called `/tmp/destination`, use the following command line:

```
sysadmin@localhost:~$ find ~ | cpio -pd /tmp/destination
```

To prevent problems with files that have white space characters (like the spacebar character) embedded in them specify the `-print0` option to the `find`command. This causes the list of files to be separated by the null character, instead of a new line character, which allows for filenames with spaces in them to be treated as a single filename (otherwise a file named `hello there` would be considered two files, one named `hello` and the other named `there`).

For the `cpio` command to process the list of null separated files, add the `--null` option. This results in a more robust version of the previous pass-through command that looks like this:

```
sysadmin@localhost:~$ find . -print0 | cpio --null -vd /tmp/destination
```

Consider This

To understand why the `cpio` command might be used for copying files from one directory to another instead of using the `cp` command recursively, consider the following advantages:

- The `cpio` command automatically preserves file attributes (metadata) like links, permissions, timestamps and ownerships. These attributes are not preserved with using the `cp` command.

- The `cpio` command also works with special files better than the `cp`command.

12.8 dd Command

The `dd` command is a utility for copying files or entire partitions at the bit level. This command has several useful features, including:

- It can be used to clone or delete (wipe) entire disks or partitions.
- It can be used to copy raw data to removable devices, such as USB drives and CDROMs.
- It can backup and restore the MBR (Master Boot Record), a critical software component that is used to boot the system.
- It can be used to create a file of a specific size that is filled with binary zeros, which can then be used as a swap file (virtual memory).

The `dd` command uses special arguments to specify how it will work. The following illustrates some of the more commonly used arguments:

- `if` - the input file to be read.
- `of` - the output file to be written.
- `bs` - the block size to be used. By default, the value is considered to be in bytes. Use the following suffixes to specify other units: `K`, `M`, `G`, and `T`for kilobytes, megabytes, gigabytes and terabytes.
- `count` - the number of blocks to read from the input file.

In the following example, a file named `/tmp/swapex` is created with 500 "one megabyte" size blocks of zeros:

```
sysadmin@localhost:~$ dd if=/dev/zero of=/tmp/swapex bs=1M count=500
500+0 records in
500+0 records out
524288000 bytes (524 MB) copied, 0.825745 s, 635 MB/s
```

No block size or count needs to be specified when copying over entire devices. For example, to clone from one hard drive (`/dev/sda`) to another (`/dev/sdb`) execute the following command:

```
sysadmin@localhost:~$ dd if=/dev/sda of=/dev/sdb
```

The `dd` command can even be used to make an `.iso` image backup of your CDROM or DVD device. The following will take all of the data from the DVD (`/dev/dvd`) and stores the data into a local file called `dvd.iso`:

```
sysadmin@localhost:~$ dd if=/dev/dvd of=dvd.iso
```

Consider This

Device files are files used to refer to devices on the system, such as hard drives, CDROMS and partitions. The following information is provided to add clarity to the examples shown with the `dd` command:

- `/dev/sda` - A device file that typically refers to the first hard drive on the system.
- `/dev/sdb` - A device file that typically refers to the second hard drive on the system.
- `/dev/dvd` - A device file that typically refers to the first DVD drive on the system.

Chapter 12: Archive Commands

This chapter will cover the following exam objectives:

103.3: Perform basic file management

Weight: 4

Description: Candidates should be able to use the basic Linux commands to manage files and directories.

Key Knowledge Areas:

- Usage of tar, cpio and dd
  Section 12.1 | Section 12.2 | Section 12.7 | Section 12.8

Chapter 12: Archive Commands

bzip2

Compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78-based compressors, and approaches the performance of the PPM family of statistical compressors.
Section 12.4

cpio

Command used to copy files into and out of archives. GNU cpio is a tool for creating and extracting archives, or copying files from one place to another. It handles a number of cpio formats as well as reading and writing tar files.
Section 12.7

dd

Command used to copy and covert a file. The dd command is a simple, yet versatile and powerful tool. It can be used to copy from source to destination, block-by-block, regardless of the filesystem types or operating systems. A convenient method is to use dd from a live environment, as in a livecd.
Section 12.8

gunzip

Command that decompress files created by gzip, zip, compress, compress -H or pack.
Section 12.3

gzip

Command used to comperess or expand files. Gzip reduces the size of the named files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension .gz, while keeping the same ownership modes, access and modification times. (The default extension is -gz for VMS, z for MSDOS, OS/2 FAT, Windows NT FAT and Atari.)
Section 12.3

tar

Command that stores and extracts files from tape or disk archive. This is the GNU version of the tar archiving utility.
Section 12.2

xz

Command that can compress or decompress .zx or .lzma files. xz is a general-purpose data compression tool with command line syntax similar to gzip(1) and bzip2(1). The native file format is the .xz format, but also the legacy .lzma format and raw compressed

streams with no container format headers are supported.
Section 12.6