dd

What is dd?

A backup utility

Preforms a sequential, byte-for-byte copy

What can dd work with?

dd can use any file on the system

In Unix, everything is a file

This includes (but not limited to):

Files

Folders

Devices

Processes

"dd can use any file" includes any **device** on the system

Example 1

Backing up a hard drive to an image:

```
dd if=/dev/sda of=disk.img
```

First hard drive

Stands for “input file”

Disk image file

Stands for “output file”

This works with any disk, including CDs.

Example 2

Restoring an image to a hard drive:

```
dd if=disk.img of=/dev/sda
```

All I did was reverse input and output

We can use the `mount` command to open this image for reading/writing

Example 3

Compressing a disk image (while making it):

```
dd if=/dev/sda | gzip -c > disk.img.gz
```

Standard output from dd is taken to the standard input to gzip (which is then written to a file)

This also works in reverse:

```
gunzip -c disk.img.gz | dd of=/dev/sda
```

Example 4

Formatting and/or writing zeros to a disk will erase data, however there is the possibility of recovery

We need to write random 1's and 0's to ensure data cannot be recovered

```
dd if=/dev/random of=/dev/sda
```

Using Linux's random number generator, this will securely erase the entire disk

Example 5

What if the partition table is corrupt, and you need to restore partitions?

If you know where the partitions are, you can clear out the table, then restore the partition

```
dd if=/dev/zero of=/dev/sda bs=1 count=512
```

This will clear the partition table, and allow you to use a utility (such as fdisk) to restore the partitions

Conclusion

dd is a simple tool with a simple job

Thanks to Linux, we can use it for a variety of tasks

If we tie dd together with other simple programs and utilities, we find a wealth of functionality hidden in a few simple tools.

Appendix A: more backup

To properly backup a partition:

- dd if=/dev/sda of=output.img bs=4k conv=noerror,notrunc

bs=4k

- Means “read 4 blocks at a time”
- Ensures we properly read blocks from the filesystem

conv=noerror

- Means “do not stop for errors”

Appendix A (cont.)

- conv=notrunc
  - Means “do not truncate output”
  - We use this to prevent dd from padding the output with extraneous zeros

Appendix B: mount

If we made a disk image of a single partition, we can use mount to load it into an empty folder just like it was a hard disk

```
mount -t <fs> -o loop partition.image mount/
```

Loop tells Linux to treat the image as a device

If we did an entire disk, we need to give mount an *offset*

This offset is set to the physical position of the partition we want in the disk.

Appendix B (cont.)

If we have a disk image of several partitions, mount will need an offset to mount one of them

fdisk -l will show where the partitions begin (marked by 'start')

fdisk lists things in blocks, but mount's offset is in bytes

512 bytes per block, so multiply the start number by that to get the partition's offset

The final command

```
mount -t <fs> -o loop,<offset> image.img mount/
```

ubuntu

Appendix C: using bs

Normally, we specify bs=4k (or 4096)

Different hardware will have different optimal block sizes, but we can easily test that

```
dd if=/dev/zero of=<destination> count=10
  bs=<blocksize>
```

This will count 10 blocks, then stop and report the transfer speed

You can use this to compare the write speed of different block sizes

dd

What is dd?

A backup utility

Preforms a sequential, byte-for-byte copy

What can dd work with?

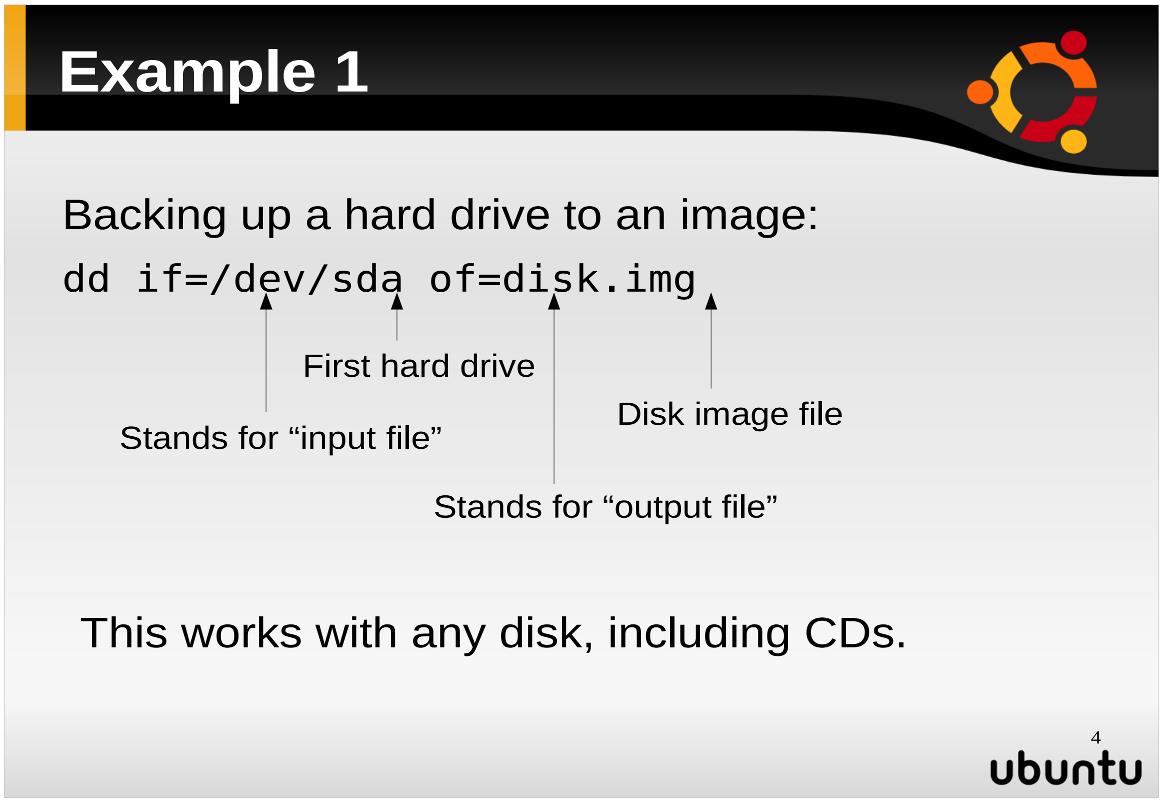
dd can use any file on the system

In Unix, everything is a file

This includes (but not limited to):

- Files
- Folders
- **Devices**
- Processes

“dd can use any file” includes any **device** on the system



Example 1

Backing up a hard drive to an image:

```
dd if=/dev/sda of=disk.img
```

First hard drive

Stands for “input file”

Disk image file

Stands for “output file”

This works with any disk, including CDs.

Example 2

Restoring an image to a hard drive:

```
dd if=disk.img of=/dev/sda
```

All I did was reverse input and output

We can use the `mount` command to open this image for reading/writing

Example 3

Compressing a disk image (while making it):

```
dd if=/dev/sda | gzip -c > disk.img.gz
```

Standard output from dd is taken to the standard input to gzip (which is then written to a file)

This also works in reverse:

```
gunzip -c disk.img.gz | dd of=/dev/sda
```

Example 4

Formatting and/or writing zeros to a disk will erase data, however there is the possibility of recovery

We need to write random 1's and 0's to ensure data cannot be recovered

```
dd if=/dev/random of=/dev/sda
```

Using Linux's random number generator, this will securely erase the entire disk

Example 5

What if the partition table is corrupt, and you need to restore partitions?

If you know where the partitions are, you can clear out the table, then restore the partition

```
dd if=/dev/zero of=/dev/sda bs=1 count=512
```

This will clear the partition table, and allow you to use a utility (such as fdisk) to restore the partitions

Conclusion

dd is a simple tool with a simple job

Thanks to Linux, we can use it for a variety of tasks

If we tie dd together with other simple programs and utilities, we find a wealth of functionality hidden in a few simple tools.

Appendix A: more backup

To properly backup a partition:

```
dd if=/dev/sda of=output.img bs=4k
   conv=noerror,notrunc
```

bs=4k

- Means “read 4 blocks at a time”
- Ensures we properly read blocks from the filesystem

conv=noerror

- Means “do not stop for errors”

Appendix A (cont.)

- conv=notrunc
  - Means “do not truncate output”
  - We use this to prevent dd from padding the output with extraneous zeros

Appendix B: mount

If we made a disk image of a single partition, we can use mount to load it into an empty folder just like it was a hard disk

```
mount -t <fs> -o loop partition.image mount/
```

Loop tells Linux to treat the image as a device

If we did an entire disk, we need to give mount an *offset*

This offset is set to the physical position of the partition we want in the disk.

Appendix B (cont.)

If we have a disk image of several partitions, mount will need an offset to mount one of them

fdisk -l will show where the partitions begin (marked by 'start')

fdisk lists things in blocks, but mount's offset is in bytes

512 bytes per block, so multiply the start number by that to get the partition's offset

The final command

```
mount -t <fs> -o loop,<offset> image.img mount/
```

Appendix C: using bs

Normally, we specify bs=4k (or 4096)

Different hardware will have different optimal block sizes, but we can easily test that

```
dd if=/dev/zero of=<destination> count=10
  bs=<blocksize>
```

This will count 10 blocks, then stop and report the transfer speed

You can use this to compare the write speed of different block sizes